From Descriptive Annotation to Grammar Specification

Lars Hellan
NTNU
Trondheim, Norway
lars.hellan@hf.ntnu.no

Abstract

The paper presents an architecture for connecting annotated linguistic data with a computational grammar system. Pivotal to the architecture is an annotational *interlingua* – called the *Construction Labeling* system (CL) - which is notationally very simple, descriptively finegrained, cross-typologically applicable, and formally well-defined enough to map to a state-of-the-art computational model of grammar. In the present instantiation of the architecture, the computational grammar is an HPSG-based system called *TypeGram*. Underlying the architecture is a research program of enhancing the interconnectivity between linguistic analytic subsystems such as grammar formalisms and text annotation systems.

1 Introduction

This paper advocates the view that all aspects of descriptive, theoretical, typological, and computational linguistics should hang together in overall precisely defined networks of terminologies and formalisms, but flexibly so such that each field can choose suitable formats, and different traditions can maintain their preferred terminologies and formalisms. Terms and symbols used for linguistic annotation are central in this enterprise, and the paper describes an algorithm by which a code suitable for sentence level annotation can be aligned with a system of attribute-value matrix (AVM) representations. An aim for further development is a similar alignment for PoS/morpheme annotation symbols.

The alignment described has as its theoretical and computational reference point an HPSG-based system, where, aside from AVMs, *types* play a crucial role. Most likely, alignment architectures with similar capacities to the one here described can have other formal frameworks integrated. For such alternatives the present system may serve as a roadmap, and hopefully more: the architecture is sought to be modular such that parts of it – such as the formal framework, or an annotation tag system - can be replaced while keeping other parts constant. At the present point, however, this is a demonstration tied to unique choices for each module in the architecture. It serves as a feasibility demonstration of the design as such, and equally much to motivate the specific annotation code presented, which is pivotal to the system as a whole.

This paper has two parts. The first part presents the sentence-level annotation code. It consists of strings of labels (connected by hyphens) where each label represents a possible *property of a sentential sign*, such as, e.g., 'has Argument structure X', 'has Aspect Y', 'has a Subject with properties Z', 'expresses situation type S', etc. The construction type specification in (1) is a first illustration of the code:

(1) `v-tr-suAg_obAffincrem-COMPLETED_MONODEVMNT`
(Ex.: English: *the boy ate the cake*)

This reads: the sign is headed by *verb*; its syntactic frame is *transitive*; it has a Subject (su) whose thematic role is *agent*, and an Object (ob) whose thematic role is *incrementally affected*; its aspectual type is characterized as a combination of *completed* and *monotonic development*.

Expressions like that in (1), characterizing a sentence from its 'global' perspective, are referred to as *templates*. The code is flexible in having no upward bound on the number of labels used in a template, and expressive in that each label represents a *statement* about some part or aspect of the sign. The code as such will be referred to as the **Construction Labeling** (***CL***) system; see section 2.

The circumstance that each individual label has the logic of a statement, is essential to the transparency of the code. This propositional character of a label also opens for the alignment of CL with a formal grammar system, which is addressed in the second part of the paper. Here we show how templates can be linked to AVMs, like the template in (1) to an AVM like (2) (in mixed HPSG/LFG style),

(2)

$$\begin{bmatrix} \text{HEAD verb} \\ \text{GF} \begin{bmatrix} \text{SUBJ} \left[\text{INDX } \boxed{1} [\text{ROLE agent}]\right] \\ \text{OBJ} \left[\text{INDX } \boxed{2} [\text{ROLE aff-increm}]\right] \end{bmatrix} \\ \text{INDX ref-index} \\ \text{ASPECT completed} \\ \text{ACTNTS} \begin{bmatrix} \text{ACT1 } \boxed{1} \\ \text{ACT2 } \boxed{2} \end{bmatrix} \\ \text{SIT-TYPE monotonic\_development} \end{bmatrix}$$

and in such a way that each individual label in the template can be seen as inducing its specific part of the AVM, as informally and partially indicated in (3):

(3)

v - - - $\left[\text{HEAD verb}\right]$

tr - - - $\begin{bmatrix} \text{GF} \begin{bmatrix} \text{SUBJ} \left[\text{INDX } \boxed{1}\right] \\ \text{OBJ} \left[\text{INDX } \boxed{2}\right] \end{bmatrix} \\ \text{ACTNTS} \begin{bmatrix} \text{ACT1 } \boxed{1} \\ \text{ACT2 } \boxed{2} \end{bmatrix} \end{bmatrix}$

suAg - - - $\left[\text{GF} \left[\text{SUBJ} \left[\text{INDX} \left[\text{ROLE agent}\right]\right]\right]\right]$

obAffincrem - - - $\left[\text{GF} \left[\text{OBJ} \left[\text{INDX} \left[\text{ROLE aff-increm}\right]\right]\right]\right]$

Thus, while the labels have a descriptive transparency essential to the descriptive functionality of the over-all code, this transparency can be 'cashed out' also in the definition of a linking between CL and grammar formalisms like that illustrated in (2) and (3). Section 3 describes a possible architecture for achieving this, centered around the computational grammar *TypeGram*.

2 Construction Labeling

In its first development, the coding system has been based on two typologically very diverse languages: *Norwegian*, and the West African language *Ga*. An overview of the system is given in (Hellan and Dakubu 2010). The end product of its application to a language is called a *construction profile* of the language, abbreviated its *c-profile*. This is an assembly of between 150 and 250 templates encoding the span of variation offered by the language in a fixed number of respects, in a code immediately comparable to c-profiles of other languages. A c-profile for both Ga and Norwegian is given in (Hellan and Dakubu op. cit.); see also (Hellan and Dakubu 2009, Dakubu 2008, Hellan 2008).

The typical method of establishing c-profiles is through *paradigm building*, where, based on one sentence of the language, one establishes the various paradigms relative to which the sentence instantiates choices, and supplements these paradigms with paradigms spun out of other sentences or constructions, ultimately establishing a full network of construction types for the language relative to the discriminants selected. ('Construction' is here used in a theory neutral way.)

The creation of c-profiles is obviously an incremental process, both in the building of templates instantiating possibilities defined by the range of discriminants recognized at any point, and in extending this range reflecting new phenomena and new languages investigated. Thus, while the stage referred to above reflects in depth work on Germanic and Kwa, significant enhancements are currently made through work on Ethio-semitic (especially through the study (Wakjira, to appear) on Kistaninya), Bantu, Indic, and other language groups, mostly not yet having achieved full c-profiles.

Although presentable as networks, in normal displays c-profiles are given as *lists*, with strict principles of ordering. Some c-profiles are also entered in the TypeCraft database (http://www.typecraft.org/), where one can search according to any labels serving as constituents of templates. At present, the number of labels employed in the code is about 40 for valence types, 90 for specifications relating to the syntactic form of specific constituents, 40 for thematic roles of specific constituents, 20 for aspect and Aktionsart values, and 60 for situation types. For valence and grammatical functions, language and framework independence in the code is made possible due to considerable agreement across traditions, whereas for participant roles and situation types, there is much less of a consolidated basis, and in these areas code development and evaluation is still a primary issue.

3 TypeGram

TypeGram is in most respects a normal HPSG-based computational grammar built on the LKB platform (Copestake 2002). Crucial to the present discussion, it has some components designed for linking it up with the CL code, which makes it possible for it to

- provide an *AVM display* of any CL template (like (2) above, for (1));
- provide a basis for a *rapid development of a parsing grammar* for any language for which a c-profile has been created;

- provide an *intermediate parsing facility* for sentences of any language even when no grammar specific to the language has been created, as long as the language has been assigned a c-profile.

We will refer to the 'basic' part of TypeGram as its *Core*. Relative to current grammar formalisms using AVMs, such as *LFG* and *HPSG* (cf. Bresnan 2001, Butt et al. 1999, Pollard and Sag 1994), the TypeGram Core borrows from LFG an inventory of *grammatical functions*, and from HPSG the use of *types*, and a design by which all components of a grammar are encoded in AVMs. Unlike most computational grammars, the Core defines analyses for phenomena not restricted to one language, but for the union of all languages for which c-profiles have been defined. (In this respect it resembles the *HPSG Grammar Matrix* ('the Matrix' - see Bender et. al, and http://www.delph-in.net/matrix/); we comment on its relationship to this system below.) The mediation between the Core and the c-profiles is induced by special type files:

- one file for each c-profile (of which there are currently three, for Ga, Norwegian and Kistaninya)

- one general file, called *Labeltypes*, for defining CL labels as *types* in terms of the Core *types*.

This architecture can be summed up as follows (with 'Ga c-types' meaning 'types corresponding to the templates constituting the c-profile for Ga', and items in boldface being items defined inside the TypeGram system):

(4)

$$\begin{tikzcd}
\text{c-profile of Ga} \arrow[r] & \textbf{Ga c-types} \arrow[dr, leftrightarrow] & & \\
\text{c-profile of Norwegian} \arrow[r] & \textbf{Norw.c-typ} \arrow[r, leftrightarrow] & \textbf{Labeltypes} \arrow[dr, leftrightarrow] & \\
\text{c-profile of Kistaninya} \arrow[r] & \textbf{Kistane c-types} \arrow[ur, leftrightarrow] & & \textbf{Core}
\end{tikzcd}$$

Thus, what communicates between the *Core* and the construction specifications in the CL code is *Labeltypes*, which in turn feeds into the language specific template definition files. The latter files build *only* on *Labeltypes*, which in turn builds *only* on the *Core*. This allows for modularity: the content of the Core can be changed, e.g., to the system of the Matrix (or even an LFG-based system), without affecting the c-profiles or the c-type inventories.

We now describe possibilities offered by the architecture.

3.1 Providing AVM displays of templates

In exemplifying this function, we use a template from Ga, along with a glossed example to illustrate the construction type:

(5) `v-ditr-obPostp-suAg_obEndpt_ob2Mover-PLACEMENT`

Amɛ-wo tsɔne	**lɛ**	**mli**	**yɛlɛ**	
3P.AOR-put	vehicle	DEF	inside	yam
V	N	Art	N	N

'They put [vehicle's inside] [yam]' = 'They put yams in the lorry.'

Here the two objects represent a *Mover* (the yam) and where the Mover is finally placed (the lorry's inside). This *Endpoint* is characterized as the inside of something, where the expression of this inside is structurally like a possessive NP construction.

In the type-file 'Ga c-types', the template in (5) is turned into a grammatical type by the type definition (6) (where '**:=**' means 'is a subtype of' and '*&*' is the operation of unification*)*:

(6)
v-ditr-obPostp-suAg_obEndpt_ob2Th-PLACEMENT :=
v & ditr & obPostp & suAg & obEndpt & ob2Th & PLACEMENT.

The way in which the individual types *v*, *ditr*, *obPostp*, etc., are here *unified* to constitute a definition of the type corresponding to the full template, corresponds to the way in which, in (3), the constituent labels of the template (1) are portrayed as contributing to its full AVM.

The defining types in (6) are in turn defined in *labeltypes*, by definitions whose defining terms are in turn defined in the *Core*.

With such type definitions in the background, the template *v-ditr-obPostp-suAg_obEndpt_ob2Th-PLACEMENT* is a type recognized in the grammar. Using the *view type definition* offered in a standard LKB interface, one sees the AVM assigned to this template.

3.2 Developing a parsing grammar

Suppose that we want to develop a grammar of Ga – GaGram -, taking advantage of the type apparatus already described. (For Ga, the lexicon (Dakubu 2009) is partly informed by the c-profile and is a resource in building the lexicon of the grammar.) What is missing is defining a lexicon, inflectional rules, derivational rules and syntactic combinatorial rules. The latter is partly deducible from the constructional templates, and for templates which reflect verb subcategorization frames, lexical frame types are fairly directly derivable from the templates. What needs to be done in addition is specifying the lexical root items of Ga, and the inflectional and derivational formatives used in the language.

This 'grammar construction kit' offered by TypeGram clearly resembles the *HPSG Grammar Matrix* ('Matrix'; cf. Bender et al. 2002). It differs from the Matrix most essentially through the way in which the grammar internal specifications are 'semi-automatically' updated as the c-profile grows. This systematic linkage between a cross-linguistic descriptive classification code and a computational grammar code is not yet available in the Matrix. Nothing, though, precludes introducing the TypeGram architecture also there, in this respect.

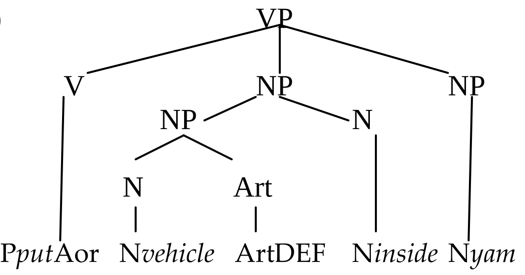
3.3 An intermediate parsing facility

TypeGram has specifications which, in addition to the above, in principle enable it to parse the Ga string in (5) – viz.,

(7) *Amɛ-wo tsɔne lɛ mli yɛlɛ*

as a structure like (8) (AVM not shown):

(8)



We may informally refer to (8) as an 'x-ray' of (7). As terminal nodes in the parse tree, it has the English glosses corresponding to the Ga roots, and functional morph glosses for the actual formatives of the Ga string. This is achieved through having as input to the parser not the string (7) itself, but the standard *gloss* associated with the string – (9a) – suitably modified to standard LKB parse input format:

(9)

a.

3P.AOR-put	vehicle	DEF	inside	yam
V	N	Art	N	N

b. V3P*put*Aor N*vehicle* ArtDEF N*inside* N*yam*

This is achieved by having the TypeGram lexicon contain all those English roots which ever appear in the glosses of Ga sentences (obviously relative to a limited, but in principle expandable corpus), and having these roots be associated with exactly the frame types which the corresponding Ga roots have relative to Ga. Thus, to produce (8), this lexicon would have to include an entry like (10) (using LKB style format), 'put' being the counterpart to *wo* in this context:

(10)

```
put := v-ditr-obPostp-suAg_obEndpt_ob2Th-
PLACEMENT & [ ORTH <“put“>,
                    ACTANTS.PRED put_rel ].
```

What this facility amounts to is a parser displaying the structure of sentences of a language for which one has designed a c-profile, but not yet a parsing grammar. It would be useful as a tool for typological comparison. To work, such a system would require a highly disciplined set of conventions for 'standard' glossing, and an interface in addition to LKB where such a glossing would be 'read in' as a string-to-parse; the latter is a facility not yet implemented (the only existing candidate interface for this purpose, to our knowledge, would be *TypeCraft* (cf. Beermann and Mihaylov 2009), while the development of the former (presumably with reference to existing glossing conventions such as the Leipzig Glossing rules, see References) would be part of the over-all initiative described at the outset.

4 Conclusion

With the Construction Labeling code and its deployment across languages as a basis, we have shown how this code can be mapped to a grammar formalism, both formally and computationally. We are thereby able to, at one and the same time, develop descriptive sentence level annotations across typologically diverse languages with a unitary code, and derive from these annotations facilities for automatic display of AVMs for any coded annotation, for rapid grammar development for the language concerned, and – so far less robustly - for intermediate 'gloss'-reflecting parsing.

We have thereby provided a system where descriptive, theoretical, typological, and computational concerns are brought together in an over-all precisely defined network of terminologies and formalisms, and flexibly so such that each field – here annotation and grammar development – have their respective suitable formats.

References

Dorothee Beermann and Pavel Mihaylov 2009. Type-Craft – Glossing and Databasing for Linguists. Proceedings of the 23rd Scandinavian Conference of Linguistics, Uppsala, Sweden, October 2008.

Emily M Bender, 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.

Joan Bresnan. 2001. *Lexical Functional Grammar*. Oxford: Blackwell.

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

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

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

Mary Esther Kropp Dakubu. 2009. Ga-English Dictionary. Accra.

Lars Hellan. 2008. Enumerating Verb Constructions Cross-linguistically. COLING Workshop on Grammar Engineering Across frameworks. Manchester. http://www.aclweb.org/anthology-new/W/W08/#1700

Lars Hellan and Mary Esther Kropp Dakubu. 2009: A methodology for enhancing argument structure specification. In: *Proceedings from the 4th Language Technology Conference (LTC 2009)*, Poznan.

Lars Hellan and Mary Esther Kropp Dakubu. 2010. *Identifying Verb Constructions Cross-linguistically*. SLAVOB series, Univ. of Ghana (http://www.typecraft.org/w/images/d/db/1_Introlabels_SLAVOB-final.pdf, http://www.typecraft.org/w/images/a/a0/2_Ga_appendix_SLAVOB-final.pdf, http://www.typecraft.org/w/images/b/bd/3_Norwegian_Appendix_plus_3_SLAVOB-final.pdf)

Carl Pollard and Ivan Sag. 1994. *Head-Driven Phrase Structure Grammar*. Chicago University Press.

Bedilu Debela Wakjira. To appear. Kistaninya Verb Morphology and Verb Constructions. PhD dissertation. NTNU.

Some web sites:

Leipzig glossing rules:

http://www.eva.mpg.de/lingua/resources/glossing-rules.php

TypeGram:

http://www.typecraft.org/tc2wiki/TypeGram

TypeCraft:

http://www.typecraft.org/

Construction Labeling site:

http://www.typecraft.org/research/projects/Verbconstructions/