DELPH-IN related activity in Trondheim, June 2011

Presented by Dorothee Beermann and Lars Hellan

DELPH-IN Summit Meeting 2011

I. *TypeCraft*

II. '*MalGram*' – extending NorSource with an error-recognizer – *NorMal*

III. An all-purpose *Construction Labeling* system, focused on argument structure, being integrated into NorSource for test-suite declaration and verb lexeme types

IV. A design experiment, joining TypeCraft and the Matrix system for a scenario of deriving grammars from flat annotation

II. MalGram

- MalGram is a cooperation with ERG where NorSource is extended in a similar way as ERG, in showing how broad-coverage implemented grammars can be augmented with an inventory of mal-rules (Schneider and McCoy 1998, Bender et al. 2004) and analogous mal-lexemes which enable precise error analysis of students' sentence composition in language arts and writing courses. Our goal in this work is to provide accurate and detailed instruction to students in response to each sentence that they write while taking these online courses.

- The ‘mal’-extension of NorSource we call *NorMal*. The first envisaged use of NorMal is in the online course *NoW*, taught at NTNU as an introduction to *Norwegian for non-Norwegian students*, in the Summer and Fall semesters 2011.
- Participants in MalGram are Lars Hellan, Tore Bruland, Elias Aamot, and Mads H. Sandøy for Norwegian, and Dan Flickinger, instrumental in providing experience and expertise from both the ERG and *The Education Program for Gifted Youth (EPGY)* at Stanford University.
- As user interface for the system we use *TypeCraft*, with the assistance of Dorothee Beermann.

- NorMal is an augmented version of NorSource, with types and rules covering specific types of mistakes.
- The files constituting NorMal are thus essentially those of NorSource, but with specific files for the mal-phenomena in addition.
- Moreover, NorMal does not use the very large noun lexicon of NorSource, and it also avoids certain rules that one does not envisage necessary in the specific learning environment of the course.
- In the TypeCraft interface, the student has an input mask into which she can write any sentence she wants of Norwegian, with the recommendation that the input be short, with a vocabulary using that of the course, and that a general spell-checker of Norwegian may well be consulted first.

Testing the string *Jeg liker du*.

- Instruction in the interface:

  Enter a sentence and press ENTER or press the Analyze button.
- **Response** produced for the sentence:

  The word "du" is marked with the wrong case, try using "deg" instead.

Recommendation, via generation from parse:

- input sentence *Jeg liker du*.
- generated sentence *Jeg liker deg*

- The *recommendation* is the string generated from the MRS of the NorMal-parsed sentence.
- Both mal-rules and mal-lexical entries introduce into the MRS exactly the same EP(s) as their ’bon’-counterparts generally introduce, whereby generation can produce well-formed strings coming very close to the intended form.
- To enable this, parsing is done in PET, and generation in LKB.

- Currently about 30 phenomena are covered by NorMal. Below is a list of error messages and examples of the types of mal-formed strings they address:
- The word "jeg" is marked with the wrong case, try using "meg" instead.
- “Du liker jeg”
- The word "og" is not the infinitival marker, try using "å" instead.
- “Jeg prøver og komme.”
- The word "å" is not a conjunction, try using "og" instead.
- “Ola å Per kommer.”
- The reflexive pronoun "seg" does not match the number and gender of the word it refers back to. Try using "meg"
- ”Jeg skammer seg.”

- The sentence lacks subject-verb inversion.
- “Imorgen jeg kommer.”
- The sentence contains an incorrect subject-verb inversion.
- “Kommer jeg snart.”
- The word "like" is in infinitive, but should be put in past or present tense.
- “Jeg like fisken.”
- The word "prøvde" is in the past tense, but should be in infinitive.
- “Jeg prøvde å gikk.”
- The word "hus" is of neuter gender, not masculine.
- “Husen er gult.”
- The adjective "gult" is conjugated as neuter gender, but modifies a masculine or feminine noun.
- “En gult bil stod her.”
- The adjective "gul" is conjugated as singular, but modifies a plural noun.
- “De gul bilene står her.”

- The verb "prøvde" must be followed by the infinitive marker "å".
- “Jeg prøvde komme.”
- There should always be a verb in the sentence. Try using "er" or "var" before the phrase "snill".
- “Hun snill.”
- Past perfective tense requires an auxiliary verb "å ha" in addition to the past participle "kommet".
- “Jeg kommet.”
- Passive mode requires an auxiliary verb "å bli" in addition to the past participle "skutt".
- “Presidenten skutt.”
- In main clauses, sentential adverbs, such as "ikke", must be placed directly after the verb, before any objects.
- “Jeg spiste fisken ikke.”
- The verb " fortærer " requires an object.
- “Jeg fortærer.”
- The verb "traff" requires a subject, like all finite verbs in Norwegian.
- “Traff Peter.”
- The verb "skammer" requires a reflexive object.
- “Jeg skammer.”

- A possessive "s" (without an apostrophe) is required after "Ola" to specifiy a possessive relation.
- “Ola hus er gult.”
- The noun following the verb "liker" should not be introduced by a preposition.
- “Jeg liker på Ola.”
- The noun following the verb "stole" should always be introduced by a preposition.
- “Jeg stoler Ola.”
- The word "sammen" should not be followed by "med" in this context.
- “Vi går sammen med.”
- The verb "oppføre (seg)" requires that the object is not followed by "selv".
- “Ola oppfører seg selv pent.”
- Countable indefinite nouns, such as "gutt", are normally preceded by a determiner.
- “Gutt sover.”
- A singular noun which is modified by an adjective, such as "snill", should have a determiner preceding the adjective.
- “Snill gutt sover.”
- A definite noun which is modified by an adjective, such as "snille", should have a determiner preceding the adjective.
- “Snille gutten sover.”
- The determiner "et" must have the same gender, number and definiteness as the noun it modifies.
- “Et mann sover.”

III. Construction labeling

The labeling code has a simple syntax and by now covers a range of construction types known cross-linguistically in the verbal domain.

The code is not tied to any particular theoretical framework, and is independent of computational implementations or applications.

However it is concise enough that one can map it to at least formalisms like those of HPSG and LFG.

It represents ‘top-down’ annotation of sentential constructions, focusing on their argument structure. It can be combined with ‘normal’ interlinear glossing:

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.’

v-ditr-obPostp-suAg_obEndpt_ob2Mover-PLACEMENT

Amɛ-wo	***tsɔne lɛ mli***	***yɛlɛ***
3P.AOR-put	[vehicle DEF inside] Ob	[yam] Ob2

- **ditr**: double object construction;
- **obPostp**: the First Object is a 'postpositional phrase', i.e., an NP with a head expressing a spatial domain (a 'locative noun' in some terminologies) relative to the item expressed in the Specifier;
- **obEndpt**: the First Object represents the Endpoint of a movement;
- **ob2Mover**: the Second Object represents the Mover of a movement;
- **PLACEMENT**: The situation type expressed is one of *placement*.

- Slot 1 consists of a label for *Parts of Speech* of the *head* of the entire construction, including the category of possible *formatives* marked on the head.
- Slot 2 consists of a label for *valency specification* - like intr (intransitive), tr (transitive), ditr (ditransitive), and varieties thereof.
- Slot 3 consists of one or more labels for specification of *syntactic constituents*, identified by their grammatical function (subject, object, etc.).
- Slot 4 consists of one or more labels for specification of *participant roles*: agent, theme, instrument etc.
- Slot 5 consists of a label for *aspect and aktionsart*, written in CAPS.
- Slot 6 consists of a label for the *situation type* of the construction, also written in CAPS.

- Albeit the design shown above defines the system as such, the system can be mapped to a system like HPSG/LKB. This goes as follows:
- Each constituent label (what comes between ’-’ or ’_’) in such a string represents a piece of a *sign*-type specification, such that for any sequence of labels, they unify into a *sign* AVM.
- For instance, for an AVM like the one on the next slide, the construction label elements indicated above it are defined as indicated on the slide following:

v-tr-suAg_obAffincrem-COMPLETED_MONODEVMNT

He ate the cake

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

v-tr-suAg_obAffincrem-COMPLETED_MONODEVMNT

v - - - [HEAD verb]

tr - - -
$$\begin{bmatrix} \text{GF} \begin{bmatrix} \text{SUBJ} \begin{bmatrix} \text{INDX} \boxed{1} \end{bmatrix} \\ \text{OBJ} \begin{bmatrix} \text{INDX} \boxed{2} \end{bmatrix} \end{bmatrix} \\ \text{ACTANTS} \begin{bmatrix} \text{ACT1} \boxed{1} \\ \text{ACT2} \boxed{2} \end{bmatrix} \end{bmatrix}$$

suAg - - - [GF [SUBJ [INDX [ROLE agent]]]]

obAffincrem - - - [GF [OBJ [INDX [ROLE aff-increm]]]]

COMPLETED_MONODEVMNT - - -

$$\begin{bmatrix} \text{ASPECT completed} \\ \text{SIT-TYPE monotonic\_development} \end{bmatrix}$$

Any full construction label is thereby defined as the unification of its constituent labels, so that type definitions have the kind of pattern instantiated below:

v-ditr-obPostp-suAg_obEndpt_ob2Th-PLACEMENT :=

v & ditr & obPostp & suAg & obEndpt & ob2Th & PLACEMENT.

This design is implemented in the experimental grammar *TypeGram* (where the AVM design shown above is also defined).

- The integration of the label system into NorSource has not gotten to the modularized stage just illustrated, rather it uses the labels defined in full in terms of the verb lexeme types as they were before 2009. Example:

  v-trObl-oblINTERR := trans-obl-interr-verb-lexeme.

- Still, the declarations thereby given for each sentence in the test suite are true of these sentences, and they correspond fully to the verb lexeme types defined in the lexicon. Currently the number of such types in NorSource is 220.

- ***Hellan, L. (2008). Enumerating Verb Constructions Cross-linguistically. COLING Workshop on Grammar Engineering Across Frameworks (GEAF). Manchester. (http://www.aclweb.org /anthology-new/W/W08/#1700).***
- ***Hellan, L. and Dakubu, M.E.K. Identifying Verb Constructions Cross-linguistically. SLAVOB series 6.3, Univ. of Ghana. (http://www.typecraft.org/tc2wiki/The_Legon_Trondheim_Linguistics_Project)***

IV. ‘From flat annotation to grammar’

- The architecture of the *Matrix* is essentially that of a ‘grammar intersection’ – a set of types and rules common to a set of grammars. The more closely related that set of languages, the larger will probably the intersection be; and in principle one may conceive of a smallest such subset, common to all languages, which ideally might be what the Matrix provides.
- The opposite perspective is that of a ‘grammar union’ – the set of types and rules formed by the *union* of the set of types and rules of individual grammars, existing together as a functioning system. If such a cohabitation were to comprise *all* grammars, we could speak of it as a *pan-grammar*.

- A first prerequisite for designing a pan-grammar would be that all languages be breakable down to a common set of POS and morphological categories, and language data be representable on a common format reflecting this common category inventory. (Of course not in the sense that all languages have identical such inventories, but that whenever categories and features are shared, they can be represented identically in the relevant system.) This category inventory would be defined in the overall type system of the pan-grammar.
- In this type system all construction types, syntactic as well as morphological, would be defined, as a library of compatible types. The grammar of any specific language would then employ a subset of these libraries, and the library would constitute a battery of hypotheses as to how strings in any language are organized syntactically and semantically.

- How it could work in principle:
- The pan-grammar consists of a type file, an l-rules file and a syntactic rules file, reflecting the type-file's definitions. The l-rules are defined in the abstract, without morphology. You start a grammar for language L by having the pan-grammar as described in full, and otherwise empty files for lexicon and i-rules. From each annotated datum (sentence) you export its words and morphemes into lexicon-, irule- and lrule files (the 'rank' of each item being clear from the annotation). On the basis of this information, you run a parse of the string using the whole battery of rules, and from that see which syntactic rules and which grammatical types (like subtypes of 'verb lexeme') are activated for the string. (To pre-restrict the number of candidate rules, to forestall processing over-load, it is of course allowed to use general knowledge of the language or language family.) 'Activated' rules and l-rules are stored in files ultimately to replace the pan-files, in the grammar of L.

- TypeCraft has a POS and gloss-tag system satisfying the above description, and its annotations can be exported in XML files, from which morphemes and words can be fed into the relevant files, with the annotated categories.
- The files of the Matrix do not yet quite constitute a ‘pan-grammar’ in coverage, but the design of its ‘Minimal grammar’ is suited for the scenario described.
- The Matrix is still in principle a ‘getting started’-facility for grammar engineers, with some procedurally-imposed standardization effects, but with the empty files being filled in by the grammarians of different languages individually and not necessarily with much coordination between them.
- But with somewhat richer phenomena libraries, and a link to an annotation system like TypeCraft as described, nothing would exclude the Matrix from being a ‘host’ of a system as outlined.

- In the meanwhile, the LKB-grammar ‘TypeGram’ with feature geometries more like that of the ESSLLI-grammars and the Sag & al. 2003 book, has been constructed, with coverage of more than the full set of construction types treated in Kroeger 2004, which is thus an experiment in ‘universal cohabitation’ of phenomena definitions.
- The current assembly of (full’) construction types from Germanic, Niger-Congo and Ethio-Semitic counts around 500 types (with particularly little overlap between Germanic and Niger-Congo).

A middle-step:

- Morpho-phonology/-graphemics is perhaps the hardest part of any grammar. In order to in the first round of constructing a grammar for L to abstract away from this level of complexity, the TypeCraft export facility allows you to, as ‘placeholders’ for the actual morphemes, using the gloss tags representing the category of the morphemes, so that for the purpose of approaching the syntactic structure of L, you can use as parse inputs simply the strings consisting of the gloss tags of the actual example (but preserving ‘rank’ information, such as what is a word or subpart of a word).
- With a certain risk of thwarting meaning, you can here also use the English glosses for content items used in the annotation, rendering the examples more transparent for cross-language comparison.
- A view of the current grammar test suites for Norwegian, Ga and Kistaninya formed in this way can be gotten at http://typecraft.org/tc2wiki/TypeGram .

Thus, for a Ga annotated construction like

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

‘You have been granted permission.’

the string presented to the parser would be

3PRFopen road 3PRFgive 2S

rather than

Á-gbele gbɛ á-ha bo

but with identical syntax and semantics assigned.