From LFG structures to dependency relations

Paul Meurer

Abstract. In this article, I describe the derivation of dependency structures from LFG analyses, with a focus on the Norwegian grammar NorGram. Although it is the f-structures that at a first glance resemble dependency structures most, I show that c-structures are the correct starting point for the conversion, and I outline a conversion algorithm that relies on information from both c- and f-structure, the projection operator, and the grammar itself. The derived dependency structures are projective with non-atomic relations, but can be converted into non-projective dependencies with atomic relations, and further into Universal Dependency-style structures. As an application, I describe how derived dependency versions of the NorGramBank gold-standard treebank are used to train dependency parsers with acceptable precision.

1 Introduction

Lexical Functional Grammar (Bresnan 2001) is a theoretically motivated grammar formalism that allows the encoding of a very rich set of grammatical information. This is exemplified by the Norwegian LFG grammar NorGram[1], which has been used to build a large treebank of automatically parsed and disambiguated sentences (NorGramBank), including a smaller gold-standard treebank of manually disambiguated analyses (Dyvik et al. 2016).

In contrast, many existing larger treebanks are manually or semi-automatically constructed,[2] and they are expressed in more light-weight and less theory-driven formalisms, such as phrase-structure trees of the Tiger treebank type, or Dependency Grammar. The latter formalism has recently gained much attention and popularity, most notably through the Universal Dependencies (UD) initiative.[3] The UD project seeks to provide dependency treebanks for many languages (currently comprising 64 treebanks for 47 languages) in a comparable way, by using 'universally' agreed-on and accepted coding guidelines and tagsets, while at the same time trying to keep a sensible balance between divergent design goals (De Marneffe et al. 2014; Nivre, Marneffe, et

1 https://clarino.uib.no/redmine/projects/inesspublic/wiki/NorGram_documentation

2 Notable exceptions are the Redwoods and similar HPSG treebanks, which are constructed in a way similar to NorGramBank, and the Alpino dependency treebank.

3 http://universaldependencies.org

The very model of a modern linguist. Edited by Victoria Rosén and Koenraad De Smedt. BeLLS Vol. 8 (2017), DOI 10.15845/bells.v8i1.1341.

al. 2016). Among those goals are ease of comprehension also for non-linguists such as language learners and engineers, on the one hand, and suitability for computer parsing with high accuracy, on the other hand. The layout and distribution format of the UD treebanks is such that they can readily be fed into a training pipeline for a statistical parser, e.g. MaltParser (Nivre, Hall, et al. 2006), MATE (Bohnet 2010) or the Stanford Neural Network parser (Chen and Manning 2014).

Even though it is still an open question how well such derived statistical parsers perform compared to hand-crafted grammars, the idea is compelling: training a statistical parser from an existing treebank is much less time-consuming than developing a broad-coverage rule-based grammar. In addition, statistical parsers tend to be more robust and operate at a much higher speed than rule-based (e.g. LFG or HPSG) grammars.[4] Even though statistical parsers cannot compete with detailed hand-crafted computational grammars in terms of depth of linguistic analysis and richness of detail, they are nevertheless potentially more suited for certain classes of applications where a fine-grained syntactic analysis is not necessary, and speed and coverage are of higher importance. Among such applications are data mining and information extraction of various kinds.

Motivated by such considerations, and the desire to create a consistently annotated, UD-compatible dependency treebank with relatively little effort, I describe in this article a conversion algorithm from LFG to dependency structures of various types, and I present the resulting dependency treebank and a spin-off product, a set of dependency parsers for Norwegian Bokmål.

Two quite similar approaches to the conversion of LFG structures into dependency structures have been described in (Øvrelid, Kuhn, et al. 2009) and (Çetinoğlu et al. 2010). Below, I will compare their approaches to the one I have chosen. The admittedly more complex task of converting a dependency treebank into a treebank of LFG structures has also been performed (Haug 2012). Crucial to the success was the availability of structural relations in the dependency structures of that particular treebank, the PROIEL treebank (Haug and Jøhndal 2012), that go beyond what is generally coded in dependency structures, namely secondary edges.

When going from LFG to dependency structures, enough structure should be available to allow the construction of the needed dependency relations. The question is mainly which part of the rich LFG structure (c-structure, f-structure, the projection relation between them) to base the conversion on, and which information to discard.

At a first glance, it is the f-structures that resemble dependency structures most. Dependency structures can roughly be seen as impoverished f-structures, where all attributes except the functional ones, corresponding to dependency relations, and all structure sharing have been removed. Both Øvrelid, Kuhn, et al. (2009) and Çetinoğlu

4 This does not apply to dependency parsers that operate in the (rule-based) Constraint Grammar framework (Karlsson 1990).

et al. (2010) use this correspondence as the starting point for their conversions. This correspondence is however not perfect; f-structure PRED values cannot easily be related to surface words (which the dependency nodes should consist of), because the projection relation is not injective.

Øvrelid, Kuhn, et al. (2009) solve this problem by introducing generic co-head edges between the surface form contributing the PRED value of the projected f-structure and other surface forms that project to the same f-structure. Çetinoğlu et al. (2010) describe a similar approach: they construct a modified f-structure, where every surface node corresponds to a proper PRED value. However, they give no detailed account of their algorithms.

In contrast, I chose a conversion that starts with the c-structure, but exploits the f-structure and the projection operator to arrive at the correct dependency relations and labels.

The dependency relations that are the result of the algorithm that will be outlined in Section 2 are peculiar in that they inherit a characteristic of the c-structures they are derived from: they are projective, which c-structures trivially are. This entails that the derived dependency relation labels are non-atomic in general; they are the concatenation of basic grammatical function relations, e.g. in the case of long-distance dependencies. An additional transformation has to be performed to make all relations basic, at the expense of projectivity, in order to arrive at traditional dependency structures. Both representations are equivalent and can be transformed into each other.

Even these non-projective dependency structures are quite different from dependency structures that adhere to the Universal Dependencies coding guidelines. Our derived dependency structures basically inherit their head-dependent relationship from the functional relations in the LFG f-structure, which for NorGram analyses entails that function words like (non-selected) prepositions, auxiliaries,[5] modals and coordinations (but not complementizers) are heads, having content words as dependents. These structures resemble quite closely the dependencies of the PROIEL–TOROT–Menotec family of treebanks[6] and the German TüBa-D/Z treebank,[7] among others (disregarding relation names), although TüBa-D/Z treats coordination differently and more in line with UD. In the UD initiative, on the other hand, a guiding principle is that heads should be content words, whereas function words modify the head words. This design decision was made to achieve a high degree of parallelism between dependency structures of different languages.

In Section 2.6, I will outline how the non-projective dependency structures derived from LFG analyses can be converted into UD-compatible structures.

5 Other LFG grammars might treat auxiliaries differently; e.g., in the English Pargram grammar, they have no PRED value on their own and merely contribute a feature to the f-structure.

6 See http://clarino.uib.no/iness

7 http://www.sfs.uni-tuebingen.de/de/ascl/ressourcen/corpora/tueba-dz.html

Applying the mentioned conversion algorithms to an existing gold-standard LFG treebank for Norwegian Bokmål, I derived a set of three dependency treebanks: one consisting of projective structures, a second having non-projective structures with atomic labels, and a third, UD-conformant dependency treebank.

All three treebanks were used to train statistical dependency parsers using the Stanford Neural Network parser framework and the MATE parser tools.

The performance of the resulting parsers is comparable to the numbers mentioned in the literature, with some interesting differences. I have, however, not tried to fine-tune the tagset and the training parameters in order to maximize performance. Training could also benefit from an improved gold-standard corpus.

In the conclusion, I briefly mention an application the trained parser has already found in the domain of quotation extraction.

2 From c-structure to dependency

As stated in the introduction, although f-structures conceptually resemble dependency structures (by interpreting sub-f-structures as dependency nodes and their PRED values as node labels, and taking f-structure attributes as dependency relations), they cannot be converted into traditional bilexical dependency structures without resorting to c-structure and projection information. There are several reasons for this.

Firstly, linear order information is not coded in f-structures, whereas the ordered nodes of a dependency tree should mirror the surface word order of the analyzed sentence. An ordering on sub-f-structures can only be imposed by relating them to nodes in the ordered c-structure via the projection operator, and it is in many cases far from obvious how this should be done. In addition, dependency node labels are exactly the surface token strings of the sentence. F-structure PRED values, on the other hand, are in most cases the dictionary entry forms of inflected surface words, or other abstractions from the surface form. Here, the correct surface word form might be recoverable by making use of the projection operator and other information coded in the internal representation of the LFG analysis. In some cases, however, there is not even an easily discernable trace of a surface form in the f-structure at all. These problems are exemplified in Figure 1, which displays a tentative dependency structure derived from an f-structure for sentence (1). The possessive *min* 'my' projects to the predicate *pro*, the selected preposition *om* 'about' is fused with the verb predicate *drømme*om* 'dream about', and the demonstrative *denne* 'this' even gives rise to two PRED values.

(1) *Min katt drømmer om dette.*
my cat dreams about this
'My cat dreams about this.'

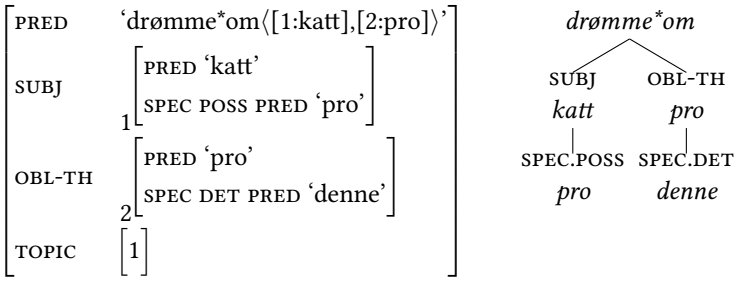


Figure 1: F-structure and derived dependency graph for (1)

For all these reasons, a more promising approach could be to derive dependency relations directly from c-structures, using f-structure information solely to construct relation labels.

2.1 The basic Lifting algorithm

In order to be able to state the algorithm that accomplishes this derivation (the Lifting algorithm), we need to recall the notion of functional head (in the c-structure!). A daughter node Y of a node X is a *functional head* (also called *f-structure head*) if Y is annotated with the equation $\uparrow{=}\downarrow$, or equivalently, Y and X share their features, or Y and X project to the same f-structure. This notion is different from the X'-theory concept of a c-structure head (e.g., N is the c-structure head of NP).

Let us first assume that every non-terminal c-structure node X has exactly one functional head Y. Under this assumption, the Lifting algorithm is easy to formulate:

Lifting algorithm, basic version.

1. Recursively replace each non-terminal node by its functional head node. In other words, lift each functional head node up to its mother node (which it replaces). Since we assume that each non-terminal node has exactly one functional head, this procedure is well-defined.

2. Label the edge between node X and daughter node Y with the f-structure path from $\varphi(X)$ to $\varphi(Y)$, where $\varphi : C \rightarrow F$ is the projection operator. If there is more than one path (because of structure sharing in the f-structure) choose the path that consists of grammatical functions (that is, contains no discourse functions like TOPIC *or* FOCUS*, but rather* SUBJ, OBJ, ADJUNCT *etc.*[8]*). If there is more than one such path, choose the shortest one. This is called the* minimal path.

8 The complete list of grammatical functions in NorGram is: SUBJ, OBJ, OBJ-TH, OBJ-BEN, OBL, OBL-TH, OBL-COMPAR, PREDLINK, COMP, XCOMP, X, ADJUNCT, NULL.

Figure 2 shows some steps in the application of the basic Lifting algorithm for example (2).[9]

(2) *Hunden sover.*
the dog sleeps
'The dog sleeps.'

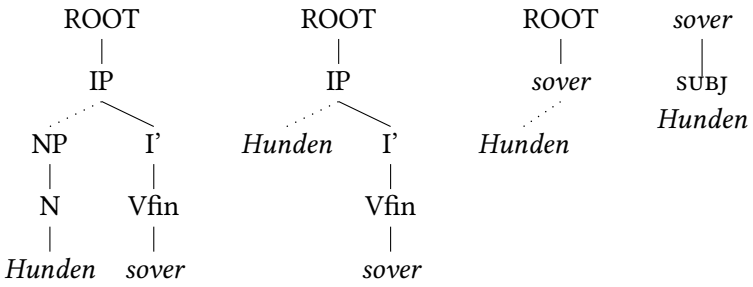


Figure 2: Steps in the application of the basic Lifting algorithm for the sentence (2)

2.2 Finding heads

In fact, all but the most basic c-structures violate the assumption of the basic lifting algorithm: a c-structure can contain nodes with multiple functional heads or without functional heads. Therefore, the Lifting algorithm has to be extended to such c-structures.

In case of multiple functional heads, the idea is to turn all but one of them into dependents, according to configurational details in the projected f-structure. We first take a closer look at c-structure co-heads.

Most c-structure terminal nodes (word forms) straightforwardly project to an f-structure whose PRED value is the associated semantic form (the base form of nouns and adjectives, and the infinitive with its subcategorization frame in the case of verbs). This is expressed via an LFG functional equation of type (3) associated to the word form, here *hund* 'dog'.

(3) (↑ PRED) ='hund'

In some cases, however, the PRED value is embedded deeper in the f-structure the surface node projects to. This is true for determiners, whose PRED value is embedded in the projected f-structure along the path SPEC DET, via equation (4). The same holds for quantifiers, which are embedded along SPEC QUANT, and possessives (SPEC POSS).

(4) (↑ SPEC DET PRED) = 'denne'

9 In this and subsequent figures, straight lines are drawn between nodes and their functional heads, whereas other c-structure edges are dotted lines.

Determiners (as well as quantifiers and possessives) are also c-structure functional heads, and their c-structure complements are f-structure co-heads, such that a c-structure fragment (5) corresponding to the phrase *denne hunden* 'this dog', where all nodes lie in the same functional domain, projects the f-structure (6). (Only the relevant parts are shown.)

(5) DP → D NP

(6)
$$\begin{bmatrix} \textsc{pred} & \text{'hund'} \\ \textsc{spec} & \begin{bmatrix} \textsc{det} & \begin{bmatrix} \textsc{pred} & \text{'denne'} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

The path from the projected f-structure to the associated PRED value I call the *embedding path* (SPEC.DET in the above example). The embedding path is empty when the PRED value is at the top level of the projected f-structure. Clearly, among several co-head nodes, we wish to turn those nodes that have a non-trivial embedding path into dependents of the node with empty embedding path (if it exists), and their relations will basically be their embedding paths. Accordingly, in the example above, the constructed relation will be (7).

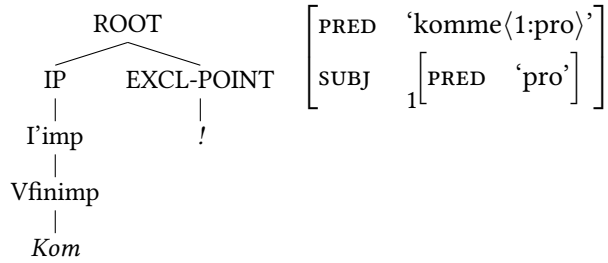
(7)

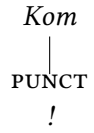

As we have seen, the embedding path of a lexical node cannot be deduced from the c- and f-structures and the projection operator alone. Rather, it has to be extracted from the functional equations attached to the lexical entry in the LFG grammar. Therefore, the details of the outlined algorithm are dependent on the grammar the sentence was parsed with.

There may also be lexical elements in the c-structure that are functional co-heads, but have no corresponding PRED value in the f-structure at all. Examples are selected prepositions and punctuation marks. Here, we arbitrarily construct an embedding path, which in the punctuation case will simply be PUNCT, as in (8, 9) for the sentence *Kom!* 'Come!'.[10] Selected prepositions will be dealt with below.

Even though coordinations are functional heads not associated with a PRED value, there is no need to give them special treatment, since they are never co-heads.

10 This example illustrates how the explicit subject information from the f-structure is lost in the conversion.

(8)


(9)


Now we are in the position to formulate the extended lifting algorithm:

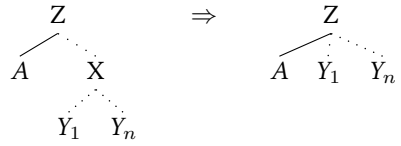
Lifting algorithm, extended version.

1a. If none of the daughter nodes $Y_1, \ldots Y_n$ of node X is a functional head, replace X by the daughter nodes as direct children of the parent Z of X, as illustrated in (10). Then proceed as before.

1b. If more than one daughter node of X is a functional head, select the node with shortest or empty embedding path as replacement for X. The remaining nodes will be treated as dependents, their relations to X being their embedding paths.

1c. As a last resort, if there is more than one such node, select the first of them as replacement.

2. Label the edge between node X and daughter node Y with the minimal f-structure path from $\varphi(X)$ to $\varphi(Y)$, concatenated with the embedding path of Y.

(10)


It is immediate from the construction that the dependency relations constructed in this way are *projective* dependencies; the sequence of words reachable from a given node along dependency arrows has no gaps, and there are no crossing edges.

Figure 3 shows the application of both *1a.* and *1b.* for sentence (11).

(11) *I dag sov noen barn lenge.*
today slept some children long
'Today some children slept long.'

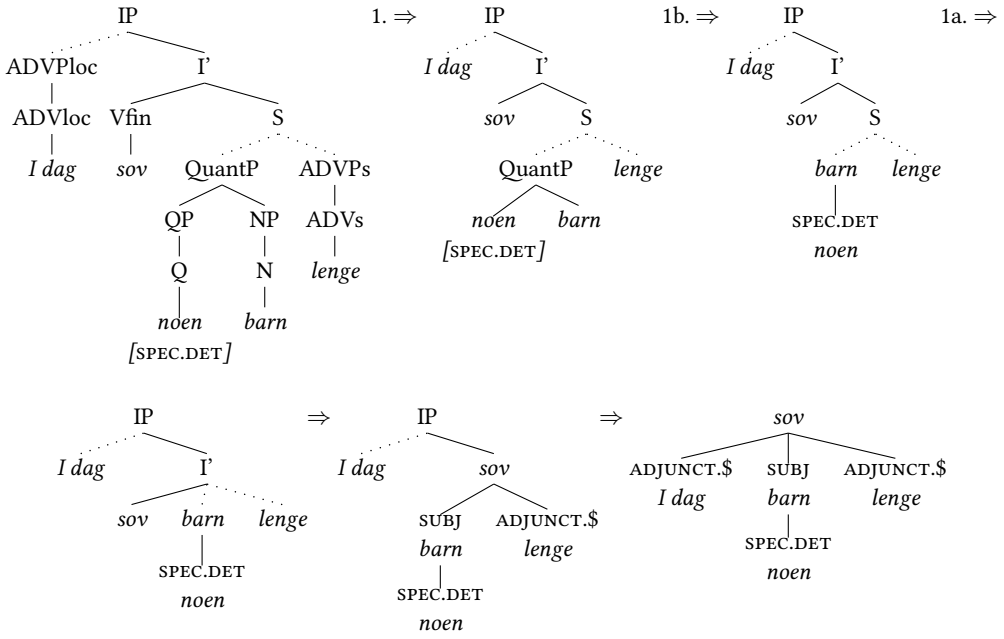


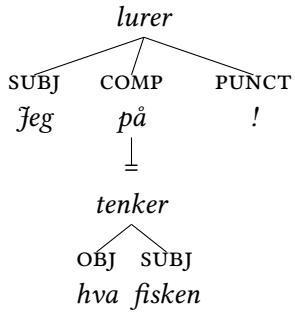
Figure 3: Steps in the application of the extended Lifting algorithm for the sentence (11). Nontrivial embedding paths are shown in brackets.

2.3 Words without PRED values

Selected prepositions have no semantic value on their own and hence do not contribute a PRED to the f-structure, and they are functional co-heads. Such a selected preposition gets an empty embedding path, which makes it the dependency head of the subordinate clause, according to rule *1c*. The relation is labeled '=', indicating that the selected preposition is not connected with a grammatical function, but merely acts as a mediator between e.g. the COMP relation and the subordinate clause, as in the analysis (13) of sentence (12).

(12) *Jeg lurer på hva fisken tenker.*
I wonder about what the-fish thinks
'I am wondering what the fish is thinking.'

(13)



Commas are treated in the same way, for similar reasons.

Complementizers including the infinitival marker *å* have no PRED value either; they merely contribute a COMP-FORM value to the f-structure. Here, the corresponding c-structure labels (Cnom, Cinf etc.) are chosen as embedding paths.

2.4 Projective vs. non-projective dependencies

As we have seen, the dependency structures derived by the Lifting algorithm are projective dependencies that potentially have compound relation labels $R = R_1.R_2 \ldots R_n.S$, consisting of the concatenation of more than one grammatical function or set inclusion R_i, and a possibly empty suffix S deriving from lexical embeddings.

This is not what dependency structures should look like. It is, however, relatively straightforward to transform such projective dependencies into non-projective dependencies without compound relation labels. The main idea is to move along the component relations of a compound relation to find the new head of a dependent. Given a relation $X \xrightarrow{R_0.R_1 \ldots R_n.S} Y$, there must also exist an atomic relation $X \xrightarrow{R_0} X_1$. The node X_1 corresponds to the c-structure surface node that gives rise to the PRED value in the sub-f-structure along R_0 of the projection of X. Then we can replace the relation $X \xrightarrow{R_0.R_1 \ldots R_n.S} Y$ by the relation $X_1 \xrightarrow{R_1 \ldots R_n.S} Y$, and by applying this process recursively we end up with a dependency structure $X_n \xrightarrow{R_n.S} Y$ (or $X_{n+1} \xrightarrow{S} Y$) without compound relation labels.

In the projective dependency structure for sentence (14) in Figure 4, there are two compound relations.[11]

(14) *Dette vet jeg ikke hva jeg skal si til.*
this know I not what I shall say to
'This I don't know what to say about.'

The second of them, '*hva* $\xleftarrow{\text{XCOMP.OBJ}}$ *skal*', is resolved by replacing it with a relation starting from the target '*si*' of the XCOMP relation, namely '*hva* $\xleftarrow{\text{OBJ}}$ *si*'. The

11 In this and the following examples, linear display mode is chosen for the dependency analyses. This mode is more suitable for longer examples, and it makes non-projectivity immediately visible through crossing edges.

other compound relation is resolved recursively, finally resulting in the relation '*Dette* $\xleftarrow{\text{OBJ:SPEC.DET}}$ *til*'.

The outlined procedure assumes that the relation $X \xrightarrow{R_0} X_1$ is unique. However, this can only be guaranteed for functional relations, in virtue of the LFG uniqueness condition. If R_0 denotes set inclusion ('$'), any of the possible relations $X \xrightarrow{\$} X$' (where X' stands for the various set members) could be taken as replacement for $X \xrightarrow{R_0.R_1...R_n.S} Y$. To avoid this problem, we label the set inclusions in the f-structure with unique subscripts.

Finally, the relations are simplified if possible; the set inclusion marker '$' is removed where it can be understood as implicit in the relation (e.g., ADJUNCT.$ is reduced to ADJUNCT since adjuncts are always set-valued), suffixes are dropped (e.g., OBJ:SPEC.DET is reduced to OBJ), and relations from lexical embeddings are simplified (e.g., SPEC.DET is reduced to DET).

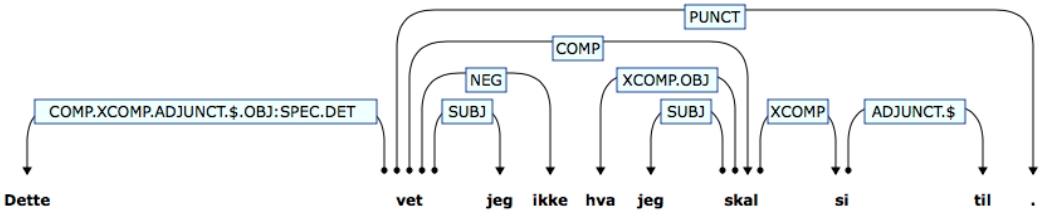


Figure 4: Projective dependency structure with compound labels for (14)

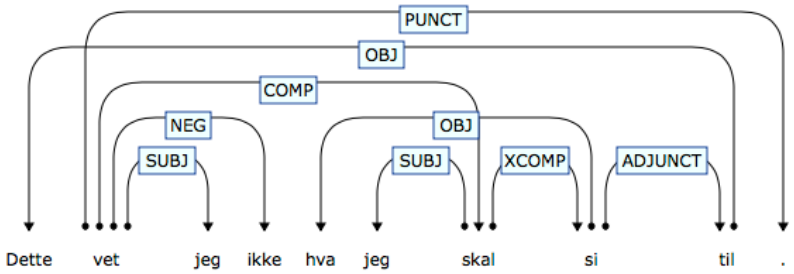


Figure 5: Non-projective dependency structure with atomic labels

2.5 Secondary edges

In some variants of Dependency Grammar, it is possible to include secondary edges, leading to structure sharing and dependency structures that no longer are trees, but directed graphs. Such structures bear an even closer resemblance to f-structures, which also are directed graphs. Secondary edges are typically used to code functionally bound arguments of the subordinate verb in raising or equi constructions, or to code the sub-

ject of a participle that modifies its own subject. In the latter case, the resulting graph is even circular. In f-structures there is also structure sharing arising from discourse functions (e.g., TOPIC, FOCUS), which could be modeled with secondary edges.

The construction of secondary edges, though not difficult in principle, is not yet covered by the described or implemented algorithms.

2.6 Conversion to Universal Dependencies

As mentioned, UD-style dependencies differ from the dependencies described in the previous sections (which I will call *LFG-style dependencies* in what follows) in the treatment of function words: whereas copula verbs, auxiliaries, modals, prepositions and coordinations are heads in the LFG-style dependencies, in the same way as they are LFG functional heads in NorGram[12] (with the exception of coordinations), function words are never heads in UD dependencies; they connect via *functional relations*, in the terminology of the UD project, to the content word. This means in concrete terms that in order to transform LFG-style dependency structures into UD dependencies, the function word head-dependent relations have to be inverted, and otherwise adapted.

This is done in an at least conceptually quite straightforward way by recursively turning a function word (copula, auxiliary, modal, non-selecting preposition) into a daughter node of the content word it heads (which amounts to reversing the head-dependent relation arrow) and turning all remaining daughter nodes of the function word into daughter nodes of the content word.

A special case are coordinations, which in the LFG-style dependencies derived from NorGram analyses are binary branching and are heading two content words (or, recursively, a content word and another binary coordination). In contrast, in UD coordinations, it is the first conjunct that is considered the head of the coordinated phrase, all other conjuncts being dependents. The coordinating conjunctions are attached to the word following them. This configuration is in accordance with the UD principle to give no function word head status; this asymmetric structure is, however, less elegant, as it does not reflect the equal semantic status of the conjuncts in the phrase.

In addition to relation edges, the relation labels too have to be adjusted in this transformation. The current version of the UD standard (UD v.2) recognizes 37 types of syntactic relations, whereas there are around 45 relation types in the LFG-style dependencies. In many cases, the UD and the LFG relations code syntactic functions at a different level of detail, and there is no straightforward mapping from LFG-style to UD-style relations. The concrete replacement of an LFG-style relation can depend on morphological features, c-structure parent labels and f-structure attribute values, in addition to the relation label itself. A typical example is the ADJUNCT relation, which translates to *nmod* if the adjunct is a noun phrase, to *amod* if it is an adjective, to *acl:relcl* if the adjunct is a relative clause, and so on.

12 One should keep in mind that other LFG grammars might treat auxiliaries differently; e.g., in the English ParGram grammar, auxiliaries only contribute with a feature to the f-structure.

The outlined derivation procedure is exemplified in Figure 6, which shows the LFG-style dependency structure for sentence (15) and the remarkably different UD-style dependency structure derived thereof.

(15) *Vi skulle ha kjørt med båt, buss eller bil.*
we should have driven with boat, bus or car
'We should have taken boat, bus or car.'

LFG allows certain types of multi-word expressions (MWEs), such as adverbials (*i dag* 'today'), complex prepositions (*ved siden av* 'next to') or named entities (*Møre og Romsdal*), to be atomic surface nodes. In contrast, MWEs are syntactically analyzed in UD; each component word represents a dependency node. UD distinguishes between three types of MWEs: fixed expressions, flat exocentric semi-fixed expressions, and endocentric analyzable compounds. Since MWEs recognized by NorGram reveal no internal structure, I treat them uniformly as fixed expressions, even though named entity MWEs arguably could be viewed as analyzable compounds with internal heads. Hence, MWEs will be annotated by attaching all non-first components to the first component via the relation *fixed*. This is shown for sentence (16) in Figure 7.

(16) *Sogn og Fjordane ligger ved siden av Møre og Romsdal.*
Sogn og Fjordane lies at the side of Møre og Romsdal
'Sogn og Fjordane is situated next to Møre og Romsdal.'

3 Training a dependency parser

The INESS NorGramBank treebank is a set of treebanks analyzed with the Norwegian LFG grammar NorGram, comprising around 5,5 million sentences (75 million words), of which 4.7 million are in the Bokmål variant of Norwegian. 28,500 of the Bokmål sentences have manually disambiguated and controlled analyses, representing the gold standard[13], whereas the analyses of the remaining sentences are disambiguated using a stochastic disambiguation module (Riezler and Vasserman 2004) that was trained on the gold standard.

As outlined in the previous sections, there are three types of dependency structures that can be derived from LFG analyses: projective and non-projective LFG-style dependencies, and UD-style dependencies. These derivations were applied to the NorGramBank gold standard, which resulted in three corresponding dependency treebanks.

In the experiment that I will describe in this section, those three treebanks were used to train statistical parsers, whose performance was then compared, against each

13 The gold standard contains only correct analyses; sentences where the grammar provides only incorrect analyses were not included.

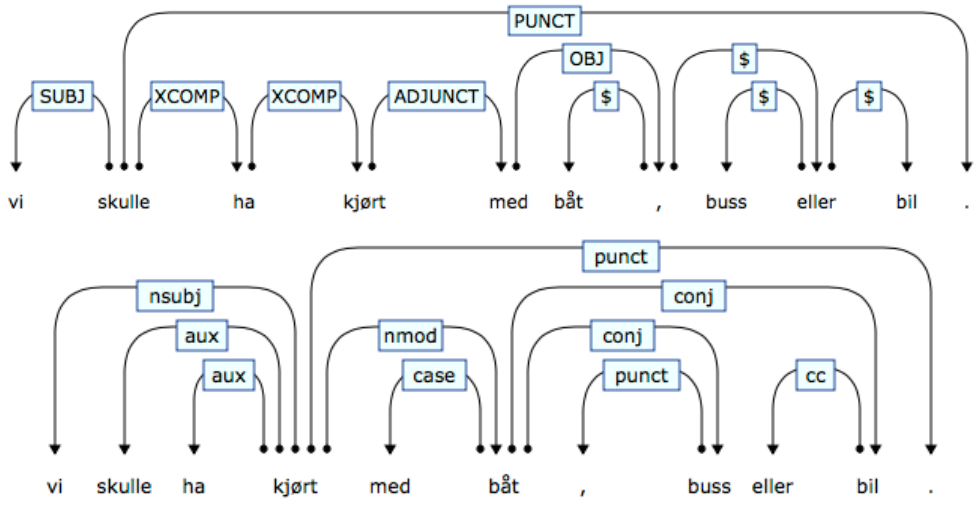


Figure 6: LFG-style and UD-style dependency structure for sentence (15)

other, and against a parser trained on the official Norwegian UD v1.4 treebank (20,000 sentences).[14]

Training of the dependency parsers was done using two different statistical dependency parser frameworks: the Stanford Neural Network Dependency Parser, and the graph-based parser from the MATE tools. As is usual, the treebank sentences were randomly divided into training, development and test sets of relative sizes 8 : 1 : 1 for each treebank. The training set had 18,859 sentences. The same training–test split was used for all three variants of the treebank. The treebanks were exported in variants of the CoNLL format, formats accepted by the parser training Java programs of the two parser frameworks. As POS tagset, the lexical categories of the c-structure nodes were chosen. Since the dependency parsers do not easily accept tokens with whitespace, NorGramBank multi-word expressions had to be split into separate tokens. As POS tags of the component tokens, the lexical category of the multi-word expression was used, extended with an '/MWE' suffix.

The parsers for the Norwegian UD v1.4 treebank were trained and tested on the training, development and test sets included in the release. For both parser frameworks, the standard training settings were used.

In addition to the training and development sets, the training algorithm for the Stanford parser also needs a word embedding file, which contains distributed representations of the words of the language. This word embedding file for Norwegian was

14 See http://universaldependencies.org

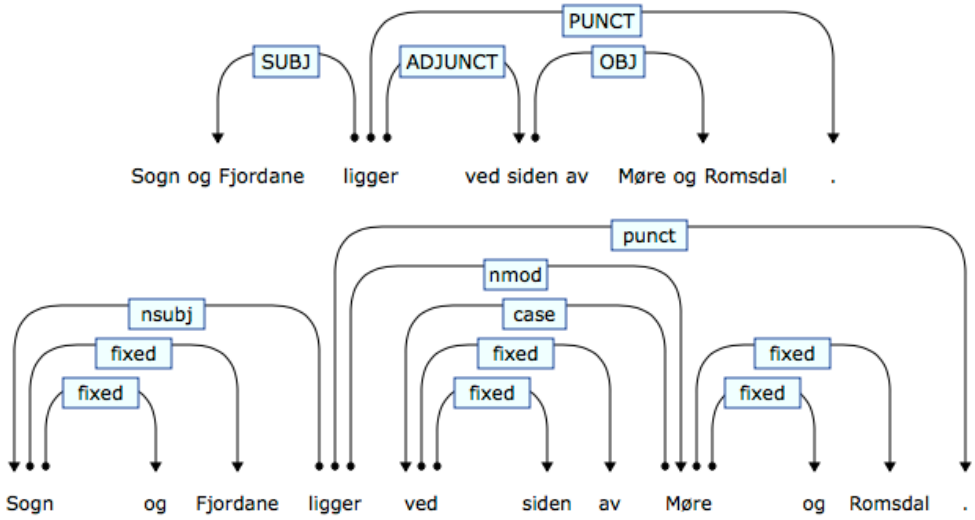


Figure 7: Treatment of multi-word expressions in LFG-style and UD-style dependencies for sentence (16)

created with the program word2vec,[15] using as input the combined word tokens of the corpora *Norwegian Newspaper Corpus* (newspaper text), *forskning.no* (popular science) and *Talk of Norway* (parliamentary debates), with more than 1.58 billion tokens altogether.[16]

The obtained precision for the parsers trained with the different treebanks and parser frameworks is given in Table 1, in terms of *Unlabeled Attachment Score* (UAS) and *Labeled Attachment Score* (LAS). In addition, the percentage of sentences with fully correct (unlabeled and labeled) attachment is given in parentheses. Both training and testing was done on pretokenized and POS-tagged input using the gold-standard POS tagset.

There is a striking difference between the parsers trained with the Stanford framework and with MATE: the MATE parsers perform much better. This is consistent with observations in the literature, where MATE is among the best performing parsers in many comparisons, e.g., Lavelli (2016) and Choi et al. (2015). However, the difference is much higher for the parsers trained on the UD v1.4 Norwegian treebank than for those trained on the LFG-derived treebanks, a fact that I cannot offer an explanation for.

15 https://code.google.com/archive/p/word2vec/

16 See http://clarino.uib.no/corpuscle

	Stanford NN		MATE	
Treebank	UAS	LAS	UAS	LAS
LFG-proj	91.67 (65.72)	89.30 (57.41)	93.81 (71.51)	90.82 (**61.58**)
LFG-nonproj	91.04 (62.44)	89.30 (57.11)	93.63 (**71.77**)	91.46 (57.54)
LFG-UD	90.71 (61.41)	88.86 (56.26)	**93.84** (70.06)	**91.83** (60.09)
UD v1.4	80.58 (38.32)	76.74 (29.04)	91.60 (56.27)	89.26 (45.49)

Table 1: Obtained precision for parsers and treebanks in terms of *Unlabeled Attachment Score* (UAS) and *Labeled Attachment Score* (LAS)

The MATE parser trained on the UD v1.4 treebank performs reasonably well; the scores are comparable to those reported by Øvrelid and Hohle (2016), who give a UAS of 91.21 and a LAS of 88.54 for MATE trained on the UD v1.2 treebank.

They note a significant difference with the scores reported by Solberg et al. (2014) for the Norwegian Dependency Treebank (NDT, the treebank that the Norwegian UD treebanks were derived from), who give a UAS of 92.84 and a LAS of 90.41 for MATE with default training settings and gold-standard POS tags. As they comment, this difference can at least partially be accounted for by the fact that the NDT annotation principles differ from those in UD in some important details: Whereas UD treats prepositions as dependents of the prepositional complement and auxiliaries as dependents of the lexical verb, they are heads in NDT.

We do not see a comparable difference in the performance of the parsers derived from the LFG non-projective dependencies and the UD-style treebanks.

The parser derived from the projective LFG-style dependencies shows a significantly lower LAS than those derived from the non-projective and the UD-style treebanks, which is probably due to the higher number of (non-atomic) relation labels in the projective case.

The scores for the MATE parsers trained on the LFG-derived treebanks may seem to be quite high; these good results should however be viewed critically. The LFG-derived UD treebank and the UD v1.4 treebank are not directly comparable, as the former contains significantly shorter sentences in average, which should make the parse process easier.[17]

It would have been interesting, and perhaps revealing, to test the LFG-derived UD parser on the UD v1.4 treebank test set, but since the treebanks use incompatible POS tags, this cannot be done.

17 The average length of the NorGramBank training sentences is 11.05, with a standard deviation of 5.15, whereas the average length of the UD v1.4 treebank training sentences is 14.81, with standard deviation 8.93.

4 Conclusion

This paper shows how three different types of dependency structures can be derived from LFG analyses, with the c-structure as starting point. All three types of dependency structures can be viewed on the web interface to the XLE parsing framework XLE-Web.[18] When a Norwegian sentence is parsed in XLE-Web, its NorGram LFG analysis is shown, alongside with a dependency structure of a chosen type.

One of the dependency parsers, namely the Stanford NN parser trained on the LFG-proj treebank, has already found a successful application in a quote extraction task (see Salway et al. 2017). In this text mining application, Norwegian newspaper articles were analyzed with the parser, and sentences that contained a speech verb having a politician's name as its subject, or a subject anaphore that could be resolved to a politician's name, were extracted. The sentence complements in the extracted constructions were the desired indirect quotes.

The conversion algorithms described in this paper, in particular the conversion to UD-style dependencies, still need some refinement: some relation types, as well as complex constructions such as comparatives and ellipsis, have not been covered yet.

The implementation of secondary edges and the corresponding conversion to UD enhanced dependencies is also left for future work. No attempt has been made to synchronize the POS tags with Universal POS tags.

5 Acknowledgments

This work and the development of the used resources have been supported by the INESS and CLARINO projects, which received partial funding from the Research Council of Norway. I want to thank the anonymous reviewers for valuable comments and suggestions. Finally, I would like to thank Helge Dyvik for many fruitful discussions, both in the context of this article, and on many other linguistic and non-linguistic topics. Talking with Helge is always a source of inspiration.

References

Bohnet, Bernd (2010). "Very High Accuracy and Fast Dependency Parsing is not a Contradiction". In: *The 23rd International Conference on Computational Linguistics (COLING 2010)*. Beijing, China, pp. 89–97.

Bresnan, Joan (2001). *Lexical-Functional Syntax*. Blackwell Publishers.

Çetinoğlu, Özlem, Jennifer Foster, Joakim Nivre, Deirdre Hogan, Aoife Cahill, and Josef van Genabith (2010). "LFG without C-structures". In: *NEALT Proceedings Series, Vol. 9*, pp. 43–54.

18 http://clarino.uib.no/iness/xle-web

Chen, Danqi and Christopher Manning (2014). "A Fast and Accurate Dependency Parser Using Neural Networks". In: *The 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*. Beijing, China, pp. 740–750.

Choi, Jinho D., Joel Tetreault, and Amanda Stent (2015). "It Depends: Dependency Parser Comparison Using A Web-based Evaluation Tool". In: *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*. Vol. 1. Bejing, China, pp. 387–396.

De Marneffe, Marie-Catherine, Timothy Dozat, Natalia Silveira, Katri Haverinen, Filip Ginter, Joakim Nivre, and Christopher D. Manning (2014). "Universal Stanford Dependencies: a Cross-Linguistic Typology". In: *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*. Ed. by Nicoletta Calzolari, Khalid Choukri, Thierry Declerck, Hrafn Loftsson, Bente Maegaard, Joseph Mariani, Asunción Moreno, Jan Odijk, and Stelios Piperidis. Reykjavik, Iceland: European Language Resources Association (ELRA), pp. 4585–4592.

Dyvik, Helge, Paul Meurer, Victoria Rosén, Koenraad De Smedt, Petter Haugereid, Gyri Smørdal Losnegaard, Gunn Inger Lyse, and Martha Thunes (2016). "NorGramBank: A 'Deep' Treebank for Norwegian". In: *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*. Ed. by Nicoletta Calzolari, Khalid Choukri, Thierry Declerck, Marko Grobelnik, Bente Maegaard, Joseph Mariani, Asunción Moreno, Jan Odijk, and Stelios Piperidis. ELRA. Portorož, Slovenia, pp. 3555–3562.

Haug, Dag Tryslew (2012). "From dependency structures to LFG representations". In: *Proceedings of the LFG'12 Conference*. Ed. by Miriam Butt and Tracy Holloway King. Udayana University, Bali, Indonesia, pp. 271–291.

Haug, Dag Tryslew and Marius L. Jøhndal (2012). "Creating a Parallel Treebank of the Old Indo-European Bible Translations". In: *Proceedings of the Second Workshop on Language Technology for Cultural Heritage Data (LaTeCH 2008)*. Ed. by Caroline Sporleder and Kiril Ribarov. Marrakech, Morocco, pp. 27–34.

Karlsson, Fred (1990). "Constraint Grammar as a Framework for Parsing Unrestricted Text". In: *Proceedings of the 13th International Conference on Computational Linguistics*. Vol. 3. Helsinki, Finland, pp. 168–173.

Lavelli, Alberto (2016). "Comparing State-of-the-art Dependency Parsers on the Italian Stanford Dependency Treebank". In: *Proceedings CLiC-it 2016 and EVALITA 2016*. Napoli, Italy.

Nivre, Joakim, Johan Hall, and Jens Nilsson (2006). "MaltParser: A Data-Driven Parser-Generator for Dependency Parsing". In: *Proceedings of the Fifth International Conference on Language Resources and Evaluation (ELREC 2006)*. Genova, Italy.

Nivre, Joakim, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajič, Christopher D. Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Sil-

veira, Reut Tsarfaty, and Daniel Zeman (2016). "Universal Dependencies v1: A Multilingual Treebank Collection". In: *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*. Ed. by Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Sara Goggi, Marko Grobelnik, Bente Maegaard, Joseph Mariani, Helene Mazo, Asuncion Moreno, Jan Odijk, and Stelios Piperidis. Paris, France: ELRA, pp. 1659–1666.

Øvrelid, Lilja and Petter Hohle (2016). "Universal Dependencies for Norwegian". In: *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*. Ed. by Nicoletta Calzolari, Khalid Choukri, Thierry Declerck, Marko Grobelnik, Bente Maegaard, Joseph Mariani, Asunción Moreno, Jan Odijk, and Stelios Piperidis. ELRA. Portorož, Slovenia, pp. 1579–1585.

Øvrelid, Lilja, Jonas Kuhn, and Kathrin Spreyer (2009). "Cross-framework parser stacking for data-driven dependency parsing". In: *TAL* 50.3, pp. 109–138.

Riezler, Stefan and Alexander Vasserman (2004). "Gradient feature testing and l1 regularization for maximum entropy parsing". In: *Proceedings of EMNLP'04*. Barcelona, Spain.

Salway, Andrew, Paul Meurer, and Knut Hofland (2017). "Quote Extraction and Attribution from Norwegian Newspapers". In: *21st Nordic Conference on Computational Linguistics (NoDaLiDa) short papers, forthcoming*. Gøteborg, Sweden.

Solberg, Per Erik, Arne Skjærholt, Lilja Øvrelid, Kristin Hagen, and Janne Bondi Johannessen (2014). "The Norwegian Dependency Treebank". In: *Proceedings of the Ninth International Conference on Language Resources and Evaluation (ELREC 2014)*. Reykjavik, Island.