FROM DEPENDENCY STRUCTURES TO LFG REPRESENTATIONS

Dag Haug
University of Oslo

Proceedings of the LFG12 Conference

Miriam Butt and Tracy Holloway King (Editors)

2012

CSLI Publications

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

Abstract

In this paper, we present the conversion of the PROIEL dependency treebank into LFG representations and the algorithms that were used in converting dependency structures to f- and c-structures. The source corpus has a large amount of non-projective edges, and the conversion to c-structure goes beyond previous work in providing principled representations of non-projective structures.

1 Introduction

When creating a treebank, it is necessary to select an appropriate formalism to express the annotation in. While it might seem obvious to simply pick your favoured linguistic theory, this is in practice rarely what happens. Instead, treebanks are usually expressed in either dependency grammar (DG), which despite some important work (Tesnière, 1959; Sgall et al., 1986; Mel'čuk, 1988; Hudson, 2007), has never really been developed as a unified linguistic theory; or in some phrase structure-based formalism that typically uses much flatter phrase structures than those assumed by linguists who use phrase structure-based paradigms as their theoretical framework.

There are several good reasons why this is so. One practical concern may be that it is hard to get annotators with the appropriate training for performing more theoretically motivated annotation. Second, corpus annotation inevitably needs to deal with constructions that linguistic theory has not developed analyses of, whether because they are thought linguistically uninteresting (e.g. calendar expressions) or simply because they have gone unnoticed in the literature (especially if one is dealing with a less-studied language). So a theoretically motivated corpus will require much theoretical work before the annotation can start. Third, creating a treebank is a very time-consuming task, and it is desirable that the result should be accessible to as many users as possible. The more theoretically motivated the treebank is, the less it is likely to be accessible to 'outsiders' who might, for example, have difficulties in navigating the AVMs of pure HPSG or LFG-based corpora.

But while there are good reasons why treebanks avoid theory-driven representations, this can lead to a gap between corpus linguistics and linguistic theory (Frank, 2001). One of the main uses that a treebank can offer would seem to be the testing of linguistic theories and analyses against 'real data', not just intuitions. But if the raw data that a corpus search yields is simply not compatible with the theory to be tested, hypothesis testing can be more difficult.

The solution to this dilemma is to annotate and store corpora using simplified representations, but to take care that it is possible to enrich these representations to proper, theoretically motivated structures – ideally, to different structures motivated by several different theoretical frameworks.

[†]We thank Mary Dalrymple for hints about the earlier literature, the audience at LFG12 for useful comments and Tracy Holloway King for editorial suggestions.

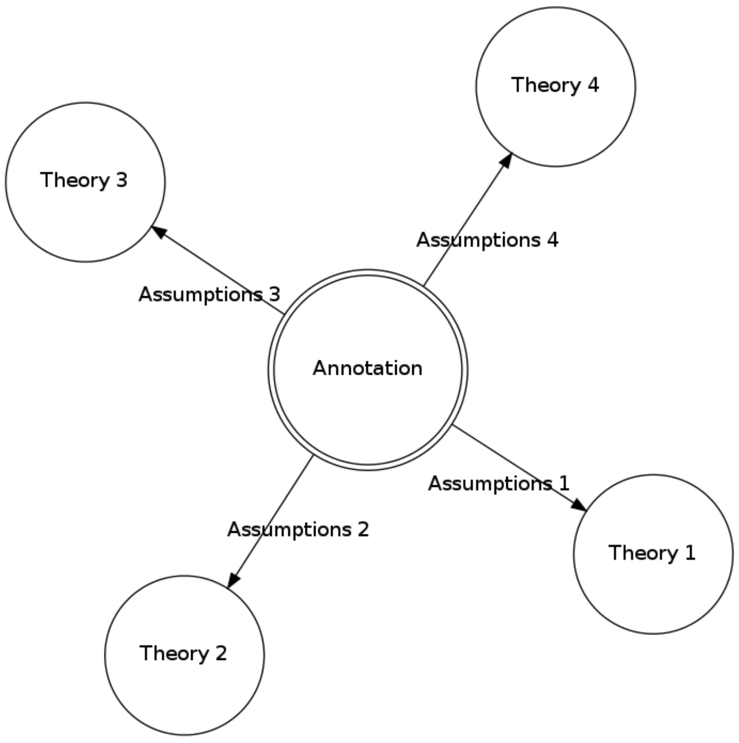


Figure 1: Linguistic annotation and theories

1.1 Theory-neutral representation

We are therefore aiming at a 'theory-neutral' representation. By theory-neutral annotation, we understand annotation that respects at least the first, and possibly the second of the following two constraints.

1. Encode enough structure to allow the reconstruction of theoretically motivated structures in the target frameworks
2. Encode no more structure than is common to the target frameworks (including structure that in some frameworks are seen as derived/secondary)

Of course, there are limits to such theory-neutrality: some theories might simply be too different. But in the ideal situation it will be possible to reconstruct full theoretical representations by supplementing the information in the corpus with the specific assumptions of each target theory. If constraint 2 is not violated, it will be possible to do this by monotonically adding information. The situation is summed up in figure 1.

When generating phrase structures from a corpus that does not have them, which is the most challenging problem in creating LFG representations out of a dependency treebank, the added theoretical assumptions will typically concern such

language	source	nonprojective	projective	% nonproj
Latin	Gallic War	1547	19086	8.1%
	Letters to Atticus	2269	22693	10.0%
	Vulgate NT	2721	65671	4.1%
	Per. Aeth.	1279	14890	8.6%
Greek	Herodotus	4137	33522	12.3%
	NT	3997	94028	4.3%
OCS	Marianus NT	1828	47719	3.8%
	Zographensis NT	26	701	3.7%
	Suprasliensis	327	6068	5.4%
Gothic	NT	1886	46884	4.0%
Armenian	NT	409	18063	2.3%
	Koriwn	48	1539	3.1%

Table 1: Projectivity in the PROIEL corpus

things as categories and X′-theory. In fact, converting a corpus can be seen as hypothesis testing: the conversion will only succeed if it is in fact possible to convert the dependency structures into phrase structures that accord with our assumptions, and failure will indicate that the data falsifies our assumptions about phrase structure.[1]

Our approach to annotation entails a different strategy for conversion than what is found in much other work. The information added during conversion is intended to embody assumptions of linguistic theories, not to be guesses about information that is lacking in the source data. With few exceptions, we are only enriching the annotation to the extent that this can be done in a deterministic way; we do not attempt to make up for missing information in the source through heuristics.

1.2 The source corpus

The corpus to be converted in this experiment is the PROIEL corpus, which consists of 447,008 words in Greek, Latin, Gothic, Armenian and Old Church Slavic (OCS). The core of the corpus is made up by translations of the New Testament in these languages (or the original, in the case of Greek), and there are also some other Greek and Latin texts. All these languages are morphologically rich, dependent-marking, non-configurational languages with a high-degree of non-projectivity, although this varies a lot between texts. As can be seen in table 1, the non-biblical texts have very high non-projectivity rates, in one case exceeding 10%. This makes the creation of motivated c-structures an interesting challenge.

The corpus is annotated with dependency structures, i.e. labelled, asymmetric relations between words, but with some changes from what is normal in DG. There

[1]It should be noted that manually annotated data often has errors and inconsitencies in it, which can also create problems for the conversion. Such annotation mistakes can and should be corrected on the source side rather than being dealt with in the conversion.

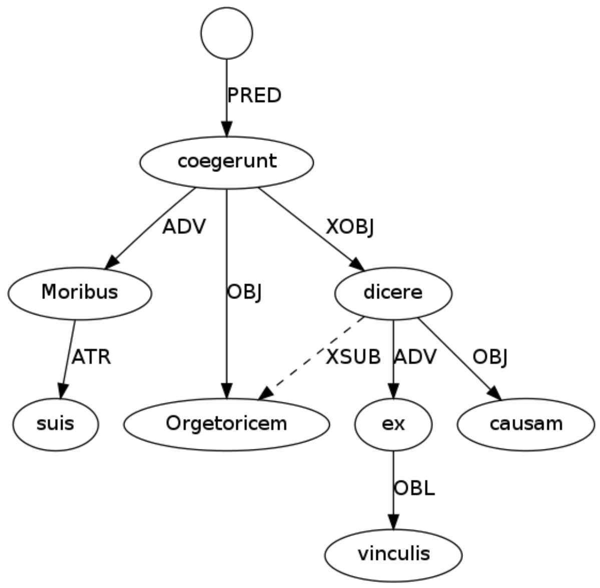


Figure 2: Object control (1)

are two problems in particular that led us to deviate from standard DG, namely structure-sharing and ellipsis.

Structure-sharing phenomena are problematic for DG because of the unique head principle which says that each word has exactly one head, ensuring that the dependency graph is in fact a tree. In the PROIEL corpus, this constraint is respected in the primary dependency graph, but structure-shared elements are related to their second head via secondary edges, as in the object control example in (1) with the associated dependency graph in figure 2.

(1)

Moribus	suis	Orgetoricem	ex
custom.PL.M.ABL	own.3.PL.M.ABL	Orgetorix.SG.M.ACC	from
vinculis	causam	dicere	coegerunt
chain.PL.N.ABL	cause.SG.F.ACC	say.PRES.INF.ACT	force.3.PL.PRF.ACT

‘Following their customs, they forced Orgetorix to speak his cause from chains.’ (Caes. Gal. 1.4.1)

Orgetoricem is both the object of the matrix verb *coegerunt* and the subject of the embedded infinitive *dicere*. Only the first function is captured in the primary dependency graph whereas the second function is represented via the dotted secondary edge labelled XSUB (external subject). Notice that the same annotation is used for functional and anaphoric control. Also no distinction is made between raising and control, i.e. whether the higher position is thematic or not. The reason is that there is simply no reliable answers to such questions for our languages at the moment. On the other hand, the annotation does distinguish between cases like (1) and those where the accusative does not have a grammatical function in the matrix clause, i.e. accusatives with infinitives. In these cases, the accusative is made the SUBJ of the infinitive (which is COMP), without any structure-sharing.

Ellipsis is problematic for DG because DG normally relies on the words of the sentence to make up the nodes of the dependency tree. But when there is ellipsis, there is structure without any word. Consider (2).

(2)

partes	tres	quarum	unam
parts.PL.F.ACC	three.F.ACC	who.PL.F.GEN	one.SG.F.ACC
incolunt	Belgae	aliam	
inhabit.3.PL.PRES.ACT	Belgian.PL.M.NOM	other.SG.F.ACC	
Aquitani	tertiam	qui	ipsorum
Aquitani.PL.M.NOM	third.SG.F.ACC	who.PL.M.NOM	himself.3.PL.M.GEN
lingua	Celtae	nostra	Galli
language.SG.F.ABL	Celt.PL.M.NOM	our.SG.F.ABL	Gauls.PL.M.NOM
appellantur			
call.3.PL.PRES.PAS			

'three parts, of which the Belgians inhabit one, the Aquitani another, and those called Celts in their own language and Gauls in ours the third.' (Caes. Gal. 1.1.1)

There are three coordinated clauses sharing the relative pronoun *quarum*, but only the first has an overt verb, *incolunt*. The others two clauses contain ellipted instances of the same verb, but this is not easy to capture in DG – there is no node of which *Aquitani* can be the subject. And there is no overt conjunction coordinating the three clauses. Our solution is to use empty nodes in such cases, as shown in figure 3. The empty nodes are typed (V for ellipted verb and C for ellipted conjunction) and bear an arbitrary, unique identifier. Via secondary edges labeled PID (predicate identity), we can also capture the fact that the empty verbal node instantiates *incolunt*.

Notice finally that this solution is in principle equivalent to that adopted in many DG corpora (notably the Prague Dependency Treebanks and corpora inspired by these) where 'invisible structure' is captured in the labels instead. We can remove empty nodes and instead make them dependents of their empty head's own head and label the resulting dependencies with the concatenation of the original label, a unique ID for the empty node, and the label that the empty node bears to its own head in the origin structure. This yields the structure in figure 4. But such structures are more difficult to work with for annotators.

The labels used in the corpus and their LFG equivalents are shown in table 2. As we can see, the labels have been chosen to match those of LFG. Still, there are a couple of differences. First, the annotation scheme does not attempt to distinguish between OBJ_θ/OBL, as that distinction is hard to draw for annotators.[2] Second,

[2]Unless it were reduced to a category difference, where all PPs are OBL and all non-accusative argument NPs are OBJ_θ – but that distinction is in any case retrievable from the morphological anno-

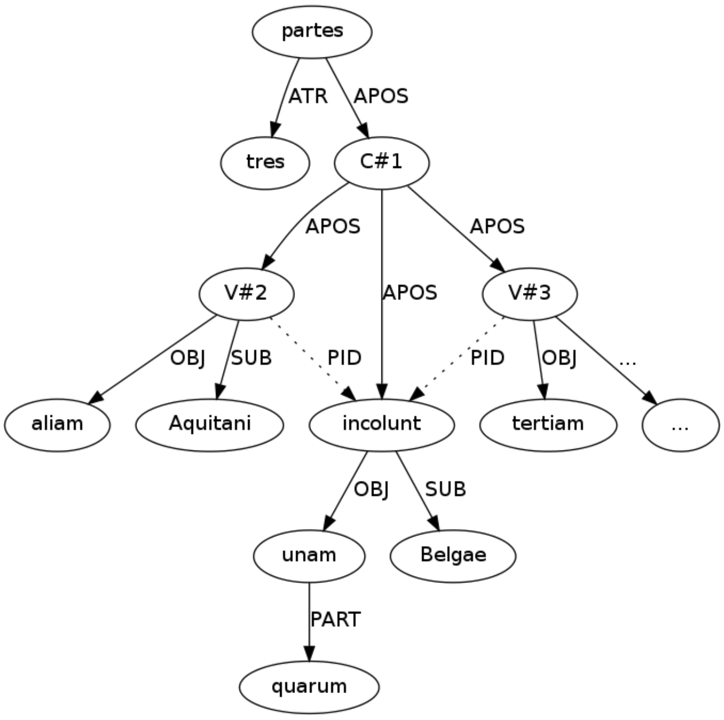


Figure 3: Ellipsis (2)

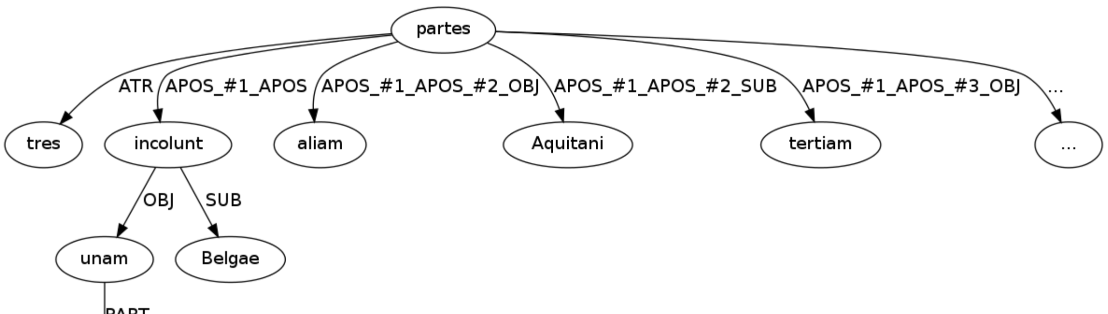


Figure 4: Ellipsis without empty nodes

Function	Label	LFG	Function	Label	LFG
Adverbial	ADV	ADJ	Oblique	OBL	OBJ_θ/OBL
Agent	AG	OBL_{AG}	Parenthetical	PARPRED	—
Apposition	APOS	ADJ	Partitive	PART	ADJ
Attribute	ATR	ADJ	Predicate	PRED	—
Auxiliary	AUX	—	Subject	SUB	SUBJ
Complement	COMP	COMP	Vocative	VOC	—
Argument of noun	NARG	OBL	Free predicative	XADV	XADJ
Object	OBJ	OBJ	Open complement	XOBJ	XCOMP

Table 2: Labels in the PROIEL corpus and their LFG equivalents

vocatives (VOC) and parenthetical predications (PARPRED) have no direct counterparts in LFG. In the conversion, these were simply ignored.[3] Predicate (PRED) and auxiliary (AUX) are used in the PROIEL corpus for functions that do not introduce an embedded layer of f-structure, but simply contribute the features of the dependent in the f-structure of the head. We will see how this works in more detail in section 2.

2 Converting to f-structures

F-structures and dependency graphs both encode labelled syntactic dependencies, so conversion is not very difficult. There is also previous work in the LFG tradition that we can lean on, notably Forst (2003).

There are two major differences between f-structures and dependency graphs that conversions typically need to deal with. First, as observed, LFG's structure-sharing runs against DG's unique head principle. This is not a problem for our conversion, since the source corpus already captures structure-sharing through secondary edges. Second, in DG every word introduces depth in the graph, whereas in LFG multiple words can contribute to the same f-structure without nesting. Here, the source corpus uses AUX and PRED for such words, so they are identifiable.

The conversion proceeds by mapping the morphological analysis of each token to a set of features and (unless the token bears the AUX function) its lemma to a semantic form. The subcategorization template for the semantic form is simply retrieved from the argument daughters in the source graph + a subject if none is present, to account for subject pro-drop.[4] This gives us f-structures for each token. In the next step, the f-structure of each token is either made the value of the LFG-equivalent of its function in the f-structure of its head (if the function is not AUX or

tation.

[3]Another simple solution would have been to generate separate f-structures for parenthetical predications.

[4]Before this is done, the lemma is checked against a stop list of impersonal words. Note that no attempt is made to account for pro-dropped objects and obliques, which are much rarer.

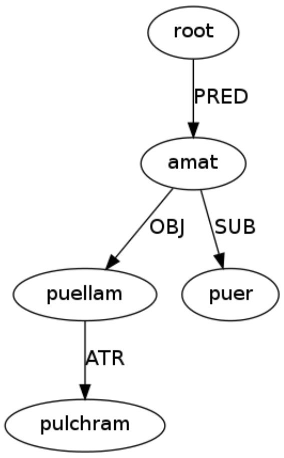


Figure 5: Dependency graph for (3)

PRED), or simply unifies with the f-structure of its head (if its function is AUX or PRED). Consider the simple example in (3).

(3)	puer	amat	puellam	pulchram
	boy.SG.M.NOM	love.3.SG.PRES	girl.SG.F.ACC	beautiful.SG.F.ACC
	'The boy loves the beautiful girl.'			

The dependency graph is given in figure 5. The f-structures (omitting NUMBER for the sake of readability) the of the words are as in (4).

(4)

$$
\begin{array}{ll}
\text{root} & {}_{r}\left[\quad\right] \\
\text{puer} & {}_{b}\left[\begin{array}{ll} \text{PRED} & \text{`BOY'} \\ \text{CASE} & \text{NOM} \\ \text{GEND} & \text{MASC} \end{array}\right] \\
\text{amat} & {}_{l}\left[\begin{array}{ll} \text{PRED} & \text{`LOVE}\ \langle \text{SUBJ, OBJ} \rangle \\ \text{PERSON} & 3 \\ \text{TENSE} & \text{PRES} \end{array}\right] \\
\text{puellam} & {}_{g}\left[\begin{array}{ll} \text{PRED} & \text{`GIRL} \\ \text{CASE} & \text{ACC} \\ \text{GEND} & \text{FEM} \end{array}\right] \\
\text{pulchram} & {}_{n}\left[\begin{array}{ll} \text{PRED} & \text{`BEAUTIFUL'} \\ \text{CASE} & \text{ACC} \\ \text{GEND} & \text{FEM} \end{array}\right]
\end{array}
$$

In step two, then, n is made the value of the ADJ function in g (embedded inside a set, since we know that (X)ADJ is set-valued), g becomes the value of OBJ in l, b becomes the value of SUBJ in l, and l and r unify, since *amat* has the PRED function. This yields (5).

(5)
$$\left[\begin{array}{ll}
\text{PRED} & \text{`LOVE } \langle\text{SUBJ, OBJ}\rangle\text{'}\\
\text{TENSE} & \text{PRES}\\
\text{PERS} & 3\\
\text{SUBJ} & \left[\begin{array}{ll}\text{PRED} & \text{`BOY'}\\ \text{CASE} & \text{NOM}\\ \text{GEND} & \text{MASC}\end{array}\right]\\
\text{OBJ} & \left[\begin{array}{ll}\text{PRED} & \text{`GIRL'}\\ \text{CASE} & \text{ACC}\\ \text{GEND} & \text{FEM}\\ \text{ADJ} & \left\{\left[\begin{array}{ll}\text{PRED} & \text{`BEAUTIFUL'}\\ \text{CASE} & \text{ACC}\\ \text{GEND} & \text{FEM}\end{array}\right]\right\}\end{array}\right]
\end{array}\right]$$

There are a couple of things to notice about the generated f-structures. First, it follows from the method that words can only contribute features to their own f-structure or that of their head – but in the latter case, they contribute *all* their features to the head. This means that it is impossible for the output structures to represent LFG's traditional account of agreement as cospecification of features in one f-structure: instead, *pulchram* bears the agreement features CASE and GEND in its own f-structure. This is not necessarily wrong (and indeed Haug and Nikitina (this volume) argue that such a theory of agreement is needed for Latin). Second, the f-structure is generated separately from the c-structure, which means that the c-structure cannot influence the f-structure. In other words, functions that are typically assigned configurationally, such as TOPIC and FOCUS cannot be accounted for.

3 Converting to c-structures

Unlike f-structures, c-structures contain information that is very different from that found in a dependency graph. This part of the conversion is therefore much more difficult, but also more interesting. As far as I know, there is no LFG work on inferring c-structures from dependencies, but there is more general work on the relationship between phrase structure grammars and dependency grammars going back to at least Gaifman (1965).

Instead of syntactic dependencies, c-structures contain information about word order, category and constituency. None of these need be present in a dependency tree. However, we can reconstruct the linear order by referring to the original string (barring tokenization differences), and in any case nodes will often bear an ordering that lets us reconstruct the original word order. Categorical information is also not too difficult to retrieve, since treebanks typically have morphological annotation.[5] In the following, we will simply assume that the words of the dependency graph are marked for their category. Finally, and this is the most crucial point, constituency and dependency are of course related. The exact relationship is the topic of discussion, which makes the conversion interesting also from a theoretical perspective.

Intuitively, we can look at constituency as combining information about dependencies and word order, i.e. a constituent is a continuous domain of words related by dependency relations. Generally, the phrasal head is also the dependency head of any words inside its phrase, but there are some common mismatches. For example, dependency analyses often (but by no means always) make functional elements such as determiners and auxiliary verbs dependents of their associated lexical element. Such differences are not really due to the difference between the formalisms, but rather to different and sometimes controversial analyses. For example, some phrase structure grammars assume that determiners take their noun as a complement, others that the determiner occurs in the specifier position of the nominal projection. We therefore consider these alterations not to be part of the conversion to c-structure. Instead they are performed in a separate step of preprocessing, where auxiliary verbs are made the head of lexical verbs (but articles remain dependents of their nouns).

In the following we will describe our conversion algorithm. We will not attempt a full-fledged formalization of the linguistic structures and the conversion between them in this context, but we will be explicit enough for it to be possible to see that the algorithm is sound and that it does not lose information, i.e. it is reversible.

3.1 What's in a dependency structure

Words are the cornerstones of our structures: they make up to nodes of our dependency graphs and the terminals of our phrase structure trees. We want the same elements to serve in both structures. More concretely, we will assume that words (and other terminals) are tuples $\langle w, i, c, r, t\rangle$ where w is the form of the word, i is the index (surface string position), c is the category, r is the syntactic function and t is a boolean flag indicating whether the 'word' is a trace. Traces will seem suspicious from an LFG perspective and we will in fact create c-structures that are trace-free, but as we will see, traces are still useful in the intermediate representations between DGs and c-structures. Notice also that we will make no use of the secondary edges in the PROIEL DGs in the conversion procedure, as we want the procedure to be applicable more generally to other corpora. In other words, the input dependency structures will respect the unique head principle. In addition, we

[5]Sometimes, the syntactic function is also necessary for category inference. For example, we assume that adjectives bearing nominal functions such as SUBJ, OBJ etc. have been nominalized and therefore have category N rather than A.

will assume that there is always a single root word, i.e. a unique word that does not have a head. Since we have ignored vocatives and parenthetical predications, this assumption holds good in the source corpus, as it does in most dependency corpora.

Dependency structures, then, will be tuples $\langle \mathcal{W}, r, f \rangle$ where $\mathcal{W}$ is the set of words, $r (\in \mathcal{W})$ is the root and f is a function $\mathcal{W} \setminus \{r\} \mapsto \mathcal{W}$ taking dependents to their heads, such that f forms a tree over $\mathcal{W}$ rooted in r. Notice that while we normally think of the labels in a dependency structure as attaching to the edges of the graph, we have here assumed that the labels attach to words. Because of the unique head principle, this makes no difference.

3.2 Order domain structures

We now introduce the notion of the order domains, a concept we have adapted from Bröker (1998), although it plays a different role in our system. The order domain $\mathcal{D}(w)$ of a node w is the largest subset of $\mathcal{W}$ such that $w \in \mathcal{D}(w)$, all words in $\mathcal{D}(w)$ are dominated (either directly or via nodes that are also in $\mathcal{D}(w)$) by w and $\mathcal{D}(w)$ is continuous, i.e. for any two words in $\mathcal{D}(w)$, all words in between are also contained in $\mathcal{D}(w)$. We will call w the head of the order domain $\mathcal{D}(w)$.

In other words the order domain $\mathcal{D}(w)$ contains w itself as well as all those of its (direct and indirect) dependents that in an intuitive sense are not ‘displaced’. This is not far from the concept of a constituent as a continuous domain of words related by dependency relations.

Let us call the set $\mathcal{O}$ of order domains of all the words in a sentence an *order domain structure*. Set inclusion is a partial order on $\mathcal{O}$ and it is easy to see that $\langle \mathcal{O}, \subseteq \rangle$ is a join semi-lattice whose top is $\mathcal{W}$, which is the order domain of r.

Furthermore, for all order domains $\mathcal{D}(w_1), \mathcal{D}(w_2), \mathcal{D}(w_3)$, if $\mathcal{D}(w_1)$ and $\mathcal{D}(w_2)$ are both supersets of the order domain $\mathcal{D}(w_3)$, then by the definition of order domains, w_1 and w_2 must both dominate w_3 in the dependency structure. Hence, since the dependency structure is a tree, either w_1 dominates w_2 or vice versa. Moreover, by the continuity of order domains, all nodes between w_1 and w_3 must be dominated by w_1 and similarly for w_2. It follows that either $\mathcal{D}(w_1) \subseteq \mathcal{D}(w_2)$ or $\mathcal{D}(w_2) \subseteq \mathcal{D}(w_1)$. This means that the order domain structure is always not just a semi-lattice, but in fact a tree in the graph-theoretical sense. Moreover, since the order domains are always continuous, there are no crossing branches in our tree. That is, order domain structures are the same kind of structures as context-free phrase structure trees over nodes that, as we noted above, are very close in conception to constituents. As such, it provides useful intermediate structure between dependency graphs and phrase structures.

Consider the constructed Latin example in (6), which has the dependency graph shown in figure 6.

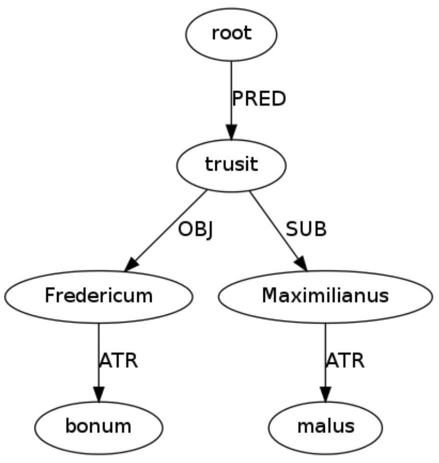


Figure 6: Dependency graph for (6) and (10)

(6)

malus	Maximilianus	trusit	bonum	Fredericum
bad.SG.NOM.M	Max.SG.NOM.M	push.SG.NOM.M	good.SG.ACC.M	Fred.SG.ACC.M

‘Bad Max pushed nice Fred.’

The order domains of the words in (6) are given in (7).

(7)

malus	{malus}
Maximilianus	{malus,Maximilianus}
trusit	{malus,Maximilianus,bonum,trusit,Fredericum}
bonum	{bonum}
Fredericum	{Fredericum,bonum}

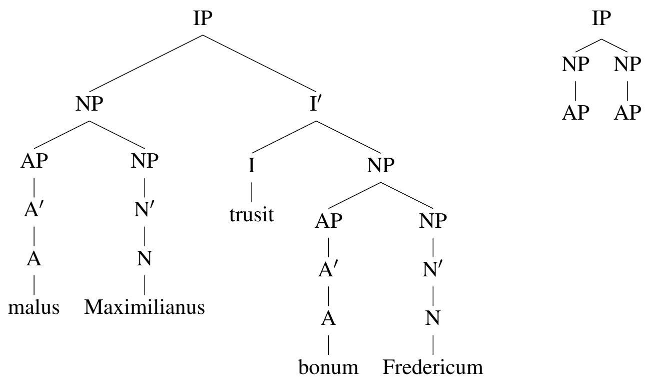
If we order these sets by set inclusion, we get the order domain structure in (8).

(8)

$$\begin{tikzcd}
 & \{\text{malus,Maximilianus,bonum,trusit,Fredericum}\} \arrow[dl, dash] \arrow[dr, dash] & \\
\{\text{malus,Maximilianus}\} \arrow[d, dash] & & \{\text{Fredericum,bonum}\} \arrow[d, dash] \\
\{\text{malus}\} & & \{\text{bonum}\}
\end{tikzcd}$$

Consider now a possible phrase structure tree for (6), along with a reduced version of this tree, containing only the (uppermost, in the case of adjunction-induced iterations) maximal projections:

(9)



We observe that the order domain structure is isomorphic to the reduced tree containing only maximal projections. Consider now (10), which gives the same example slightly altered so as to have a discontinuous object NP.

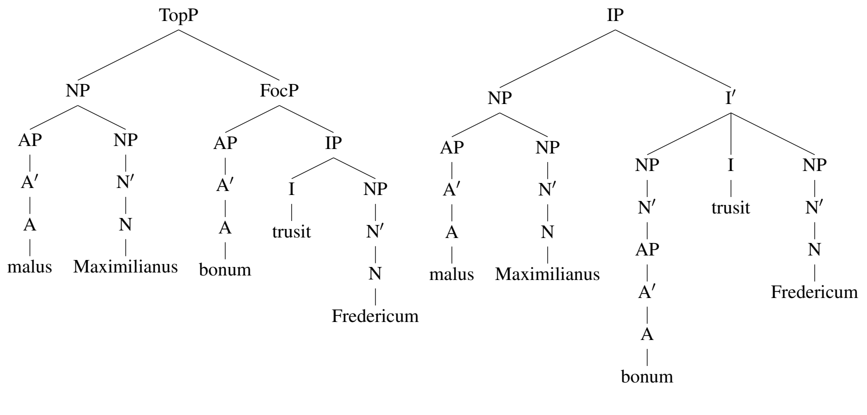
(10)

malus	Maximilianus	bonum	trusit
bad.SG.NOM.M	Max.SG.NOM.M	good.SG.ACC.M	push.SG.NOM.M
Fredericum			
Fred.SG.ACC.M			

'Bad Max pushed nice Fred.'

The dependency tree remains the same, although the indices on the words have changed. It is a theoretically disputed matter how to best represent the phrase structure of an example like (10). Two possible analyses are shown in (11). To the left is a minimalist-style analysis where the discontinuity is accounted for by assuming that the displaced adjective has moved to the specifier of a functional projection (FocP).[6] To the right is an LFG-style style analysis[7] which assumes that the displaced adjective adjoins to a headless object NP which is unified with the other object NP at f-structure.

(11)

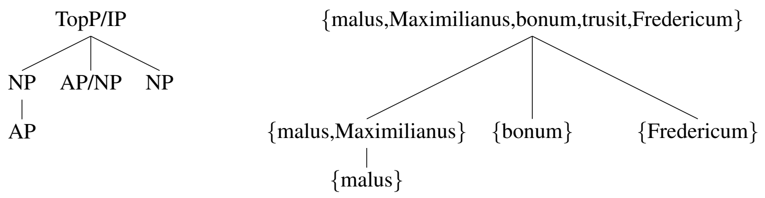


However, if we only consider the uppermost maximal projections, both these trees

[6]For an analysis of discontinuities in Latin along these lines, see Devine and Stephens (2006).

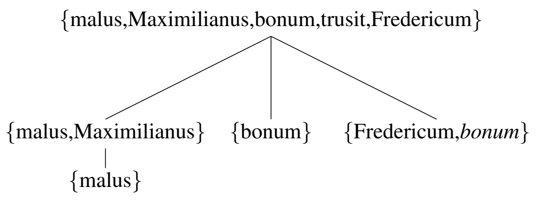
[7]This is not to say that LFG *has to* assume the rightmost structure rather than the leftmost.

are isomorphic with each other and with the order domain structure, which are both given in (12).

(12)


In a sense, then, the order domain structure represents phrase structure while abstracting away from the internal structure of projections. The internal structure of projections is of course very important in phrase structure grammars, since it determines things such as c-command relations etc. But on the other hand, assumptions about the internal structure of phrases is also where different theories differ most. Therefore, an order domain structure is as close as we can get to a proper phrase structure representation without making theory-internal assumptions. To get to proper phrase structure representations such as those in (11), we need to add those assumptions.

Before we go on to see how that can be done, we need to fix a problem with the order domain structure in (12). Although this structure can be derived from the dependency graph, there is in fact no way to go back from the order domain structure. The problem is that there is no way to retrieve the dependency of *bonum* on *Federicum*. We will solve this by enriching our order domain structures with traces. In addition to the set of nodes defined above, an order domain structure will consist of traces of all nodes that are dominated (directly or indirectly) by a word but are not in its order domain according to the previous definition. The partial order that defines the tree structure of an order domain structure will be the subset relation modulo traces. This yields (13), where traces appear as words in italics.

(13)


Now we can retrieve the dependency tree by ordering the order domains by a subset relation $\subseteq_t$ that consider traces and their overt realizations as identical.

3.3 Adding linguistic knowledge

To take order domain structures to phrase structures there are two steps we must accomplish. First, we must create a projection corresponding to each order domain. Second, we must embed these phrases in each other. To do this, we must know the rules for creating projections and combining them.

Xia and Palmer (2001) identified three questions that any conversion from dependencies to phrase structures must answer:

1. For a category X, what kind of projections can X have?
2. If a category Y depends on a category X in a dependency structure, how far should Y project before it attaches to Xs projection?
3. If a category Y depends on a category X in a dependency structure, to what position on X's projection chain should Y's projection attach?

Our answers to these questions are guided by X$'$-theory and LFG's approach to discontinuities.

1. All categories X project two levels X$'$ and XP. However, if the phrase is displaced, it will be embedded inside one or more headless projections corresponding to the path to its functional heads.
2. A dependent Y always projects to Y$'$ then YP and the YP attaches to the head's projection
3. Dependents are divided into three types using a set of handwritten rules: specifiers, modifiers and arguments. Specifiers are made sisters of X$'$ and arguments are made sisters of X. Modifiers Chomsky-adjoin to either X$'$ or XP depending on whether they are restrictive, as indicated by the dependency edge label (ATR or APOS).

The conversion of the order domain structure starts from the root, but recursively converts daughter order domains before mothers. For each order domain we first create a projection according to these rules, i.e. an X – X$'$ – XP spine, where X is the category of the order domain's head.[8] Next, if there are order domains in the tree that contain a trace of the order domain's head, we order these by $\subseteq_t$ and embed the original projection successively inside headless structures Y$'$ – YP, where Y is the category of the head of the order domain containing the trace.

Once the full projections of any daughter nodes in the order domain structure have been created, as well as the X – X$'$ – XP spine of the current node, it is necessary to embed the daughter projections in correct positions in the structure. We rely on hand-written rules to do this. A sample rule for nominal projections is given in table 3.

The rule that governs the relationship between position and grammatical function (e.g. phrase adjuncts should be non-restrictive (APOS) etc.) are for the moment hard-coded in the conversion program, so the handwritten rules only deal with restrictions on categories. The rules are tested from the 'outside in', i.e. the outermost dependents are tested for whether they can be phrasal adjuncts; if yes, the next outermost dependents are also tested and so on; when the test fails, it moves to check

[8]The empty nodes used in ellipsis and asyndetic coordination are represented as phrases without heads.

N:

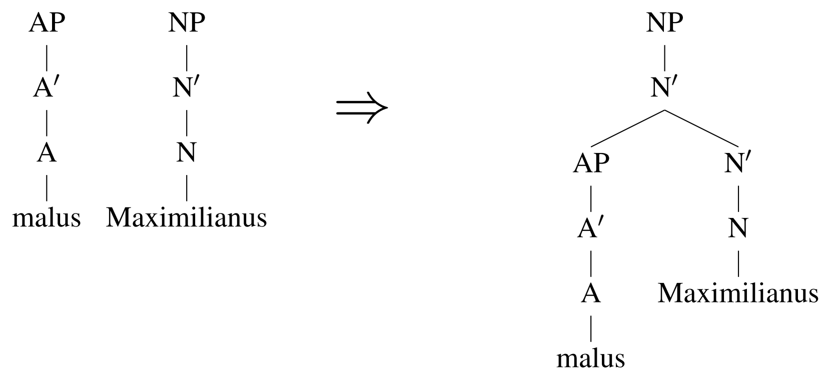
:phrase_adjuncts:	NP, AP
:specifier:	DP
:bar_adjuncts:	NP, AP
:complements:	NP, PP, AdvP, AP, CP, IP, VinfP, VptcpP

Table 3: Rules for nominal projections

the next leftmost dependent for specifier-hood; and so on, to bar-level adjunction and complementhood.

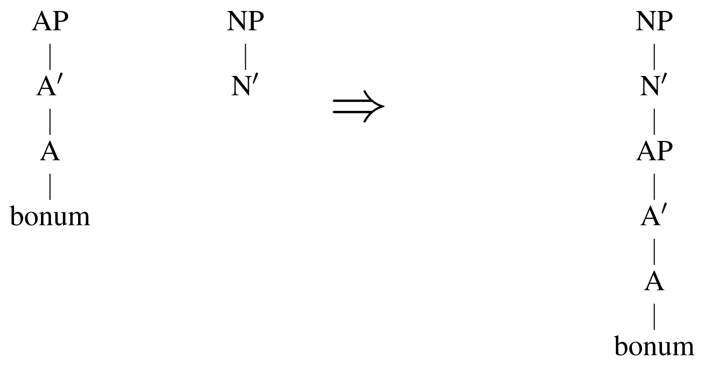
Let us see how this works for (13). In order to create a phrase structure for the top node, we must create projections from the daughter nodes. We start from the left. To create a projection for {malus, Maximilianus} we must create one for {malus} and attach it correctly in the one for {Maximilianus}. This is shown in (14). The two starting projections are determined by our X'-theoretic assumptions, and the correct combination is given by the rules in 3.

(14)



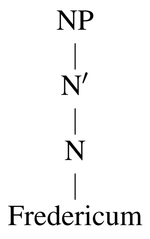
Next, we must construct a projection for *bonum*. Since there is a trace corresponding to *bonum*, we must also create a headless projection for the (possibly multiple, but in this case only one) order domain containing the trace, and embed the first one in the second. This is shown in (15).

(15)



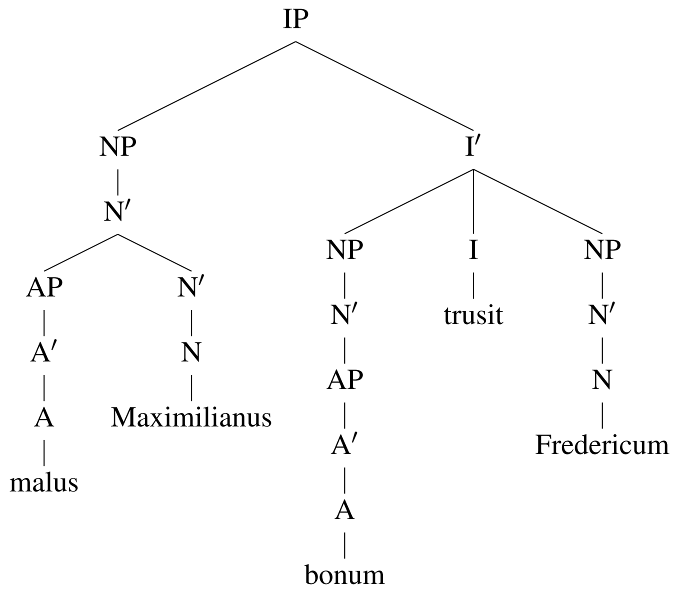
Finally, we create a projection for *Fredericum*. Since this order domain contains only one non-trace element[9] and there is no trace corresponding to *Fredericum*, this is simple:

(16)



Once we have created the projections in (14)–(16), we need to attach them in the projection of the topmost order domain, which is an IP – I′ – I spine. The result is shown in (17).

(17)



So we have arrived at a c-structure which can be motivated within LFG. Observe that in itself, this structure cannot be reverted to the order domain structure, because again there is no way to retrieve the dependency of *bonum* on *Federicum*, which should be represented by a trace in the order domain structure. For our purposes, this is not a problem, since we also generate an f-structure, from which we can see that *bonum* belongs to *Federicum*. For other applications where only c-structures are generated, other options must be considered. One, corresponding to a principles and parameters approach, would be to represent the trace in the order domain as a trace in the phrase structure. Another option is to index maximal projections with the index of the word whose projection it is. This would leave both NPs inside I′

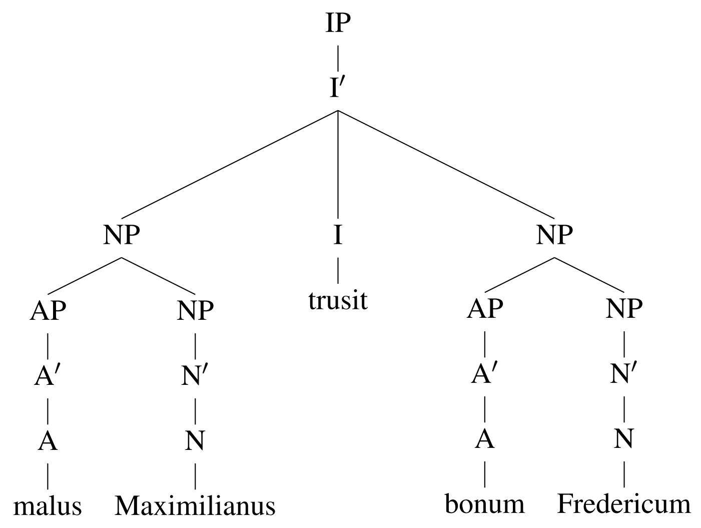
[9]Notice that if we wanted to create constituent structures with traces in them, we could treat the trace of *bonum* as projecting an empty category.

coindexed. For c-structures without such discontinuities, however, the reversion to an order domain is straightforwardly achieved by substituting for each maximal projection the set of terminals it dominates and removing all other nodes.

3.4 Evaluation

The conversion algorithm for c-structures is completely rule-based, and as such it performs no better than the rules it is fed with. But notice that for the most part, the rules do not really have the character of heuristics in the sense that they would guess the most likely alternative. Instead, they are intended as hard constraints embodying linguistic knowledge. The only exception to this is the fact that the rules are always tested from the outside in. This means that if the leftmost dependent is admissible both as a specifier and as a complement, it will always be made a specifier. The reason for this choice is that linguistic theories predict that there are elements such as *wh*-words which have to occur in specifier positions, while with other words it is often hard to see whether they are in a specifier position or not, as the semantic effects of topicalization are vague. Consider (8) again. If we assume that Latin has a non-obligatory specifier position which serves to indicate topicality (and *not* subjecthood), we could equally well assume a phrase structure as in (18).

(18)



Considerations of information structure would tell us that (18) is perhaps be more natural in an ‘all new’ context (answering *What happened?* rather than *What did Maximilian do?*). But such constraints are too vague to be of any use in conversion. This points to a more general problem with evaluating phrase structures, especially when working with ancient languages where the word order is ill understood: there is often no real gold standard to be had.

However, there is another aspect under which our conversion algorithm can be evaluated, namely its preservation of information. The algorithm does not lose linguistic information. There is no room for a formal proof here, but we have already hinted that the order domain structures, once equipped with traces, can be reverted to dependency structures. It is also possible to revert phrase structures to order domain structures.

4 Conclusion

We have seen that it is possible to convert a dependency-annotated corpora to full LFG representations, provided the original annotation is rich enough. Although dependency structures and functional structures encode very similiar kinds of information, which facilitates the conversion, this is also the part which requires the most divergences from a strict phrase structure format. The reason is that dependency grammar (at least in its usual strict form) does not allow structure-sharing. Correct transformation of the data could only be achieved because the dependency format used in the PROIEL source corpus has been extended with structure-sharing.

The conversion from dependency structures to c-structures, on the other hand, is more complicated and challenging both from a technical and a theoretical point of view. On the other hand, it does not require information beyond what is found in normal dependency structures. The conversion algorithm goes beyond previous work in conversion between dependencies and phrase structures in that it deals with non-projectivity in a principled manner, generating structures that are compatible with LFG's treatment of discontinuities.

As we noted, there are difficulties in providing an exact evaluation of the c-structure conversion, since there is room for much disagreement on what a proper c-structure of these old languages should like. Nevertheless, it is an important feature that the algorithm is reversible and does not lose linguistic information.

Finally, we hope that the converted corpora will be of use in future development of LFG grammars for these languages. As mentioned above, their phrase structure is not well understood. It is to be hoped that the converted PROIEL corpus will be a valuable resource and help towards a better understanding of both phrase structure and other aspects of the grammar of these languages.

References

Bröker, Norbert. 1998. A projection architecture for dependency grammar and how it compares to LFG. In Miriam Butt and Tracy Holloway King (eds.), *Proceedings of the LFG98 Conference*, Stanford: CSLI Publications.

Devine, A. M. and Stephens, L. D. 2006. *Latin Word Order: Structured Meaning and Information*. Oxford: Oxford University Press.

Forst, Martin. 2003. Treebank conversion – creating a German f-structure bank from the TIGER corpus. In Miriam Butt and Tracy Holloway King (eds.), *Proceedings of LFG03*, CSLI Publications.

Frank, Anette. 2001. Treebank conversion - converting the NEGRA treebank to an LTAG grammar. In *Proceedings of the Workshop on Multi-layer Corpus-based Analysis, Iasi*, pages 29–43.

Gaifman, Haim. 1965. Dependency systems and phrase-structure systems. *Information and Control* 8(3), 304–337.

Hudson, Richard. 2007. *Language Networks: The New Word Grammar*. Oxford University Press.

Mel'čuk, Igor. 1988. *Dependency Syntax: Theory and Practice*. Albany: State University of New York Press.

Sgall, Petr, Hajičová, Eva and Panevová, Jarmila. 1986. *The Meaning of the Sentence and Its Semantic and Pragmatic Aspects*. Prague, Czech Republic/Dordrecht, Netherlands: Academia/Reidel Publishing Company.

Tesnière, Lucien. 1959. *Eléments de Syntaxe Structurale*. Paris: Klincksieck.

Xia, Fei and Palmer, Martha. 2001. Converting dependency structures to phrase structures. In *Proceedings of the first international conference on human language technology research*, HLT '01, pages 1–5, Stroudsburg, PA, USA.